Отчет по 5 лабораторной работе по предмету «Типы и Структуры Данных»

# Подготовила студентка группы ИУ7-34Б

# Воякин Алексей Янович

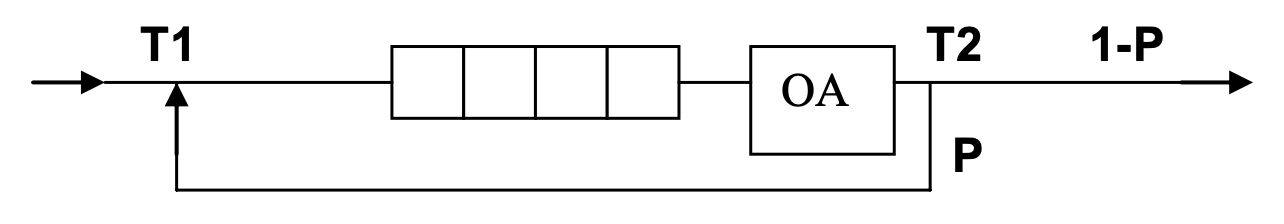
# Вариант 5

**Цель работы**

Отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценка эффективности программы (при различной реализации) по времени и по используемому объему памяти.

**Задание (Вариант 5)**

Система массового обслуживания состоит из обслуживающего аппарата (ОА) и очереди заявок



Заявки поступают в "хвост" очереди по случайному закону с интервалом времени Т1, равномерно распределенным от 0 до 6 единиц времени (е.в.). В

ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время Т2 от 0 до 1 е.в., Каждая заявка после ОА с вероятностью Р=0.8 вновь поступает в "хвост" очереди, совершая новый цикл обслуживания, а с вероятностью 1-Р покидает систему. (Все времена – вещественного типа). В начале процесса в системе заявок нет.

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок. Выдавать после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди. В конце процесса выдать общее время моделирования и количество вошедших в систему и вышедших из нее заявок, среднее время пребывания заявки в очереди, время простоя аппарата, количество срабатываний ОА. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти

**Исходные данныe**

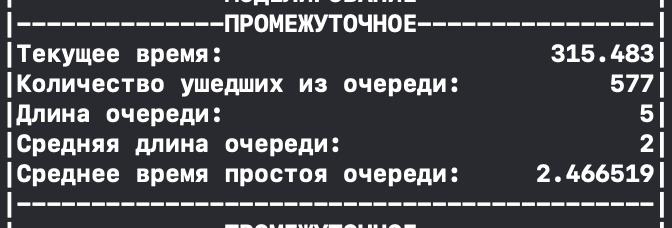
Исходные данные в данной программе не требуются, для изменения параметров следует изменить данные в файле constants.h.

**Выходные данные**

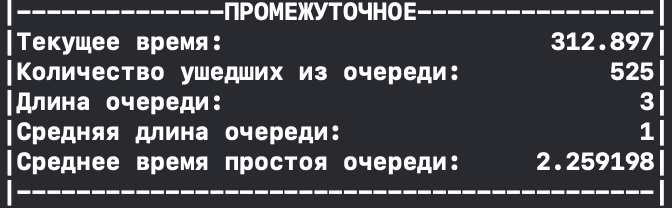
После обслуживания каждых 100 заявок выводится информация о текущей и средней длине очереди, а в конце процесса - общее время моделирования и количество вошедших в систему и вышедших из нее заявок, данные о погрешностях и некоторые другие требуемые сведения о работе программы.

**Интерфейс программы**

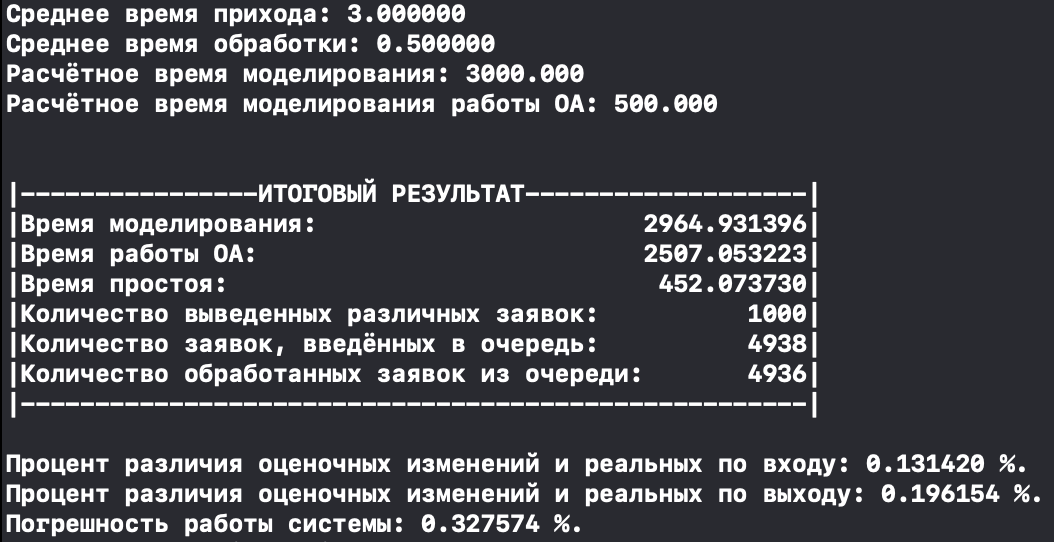
Данные об очереди при обработке первых (и каждых последующих) ста заявок первого типа. Реализация на массиве.



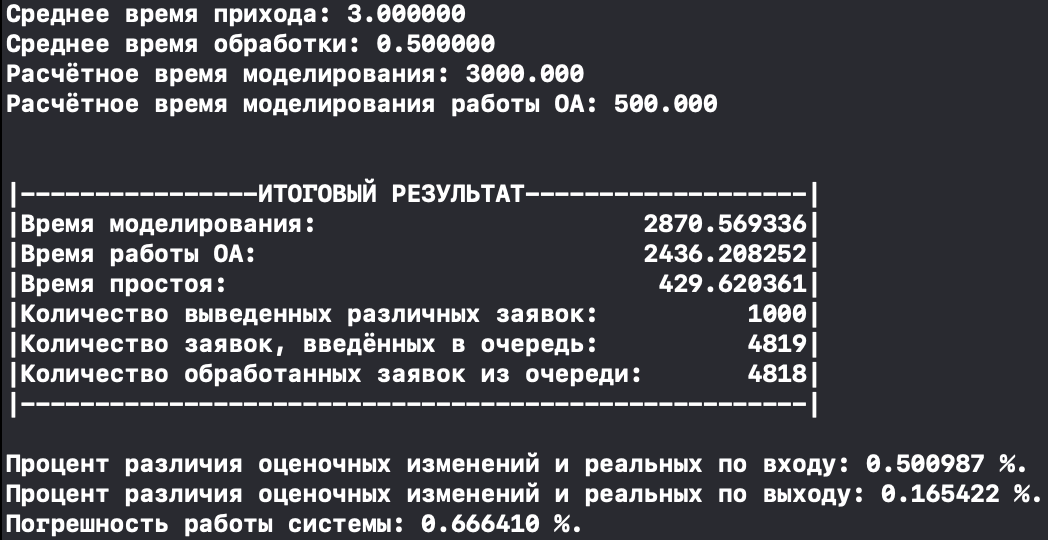
Реализация на списке.



Итоговый результат на массиве:



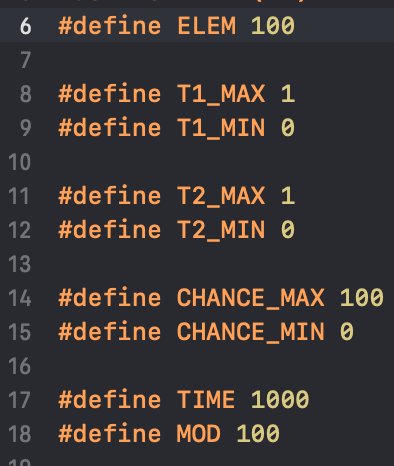
Итоговый результат на списке:



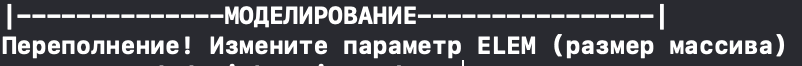
**Аварийные ситуации**

1. Переполнение при реализации массивом

Пример переполнения: параметры

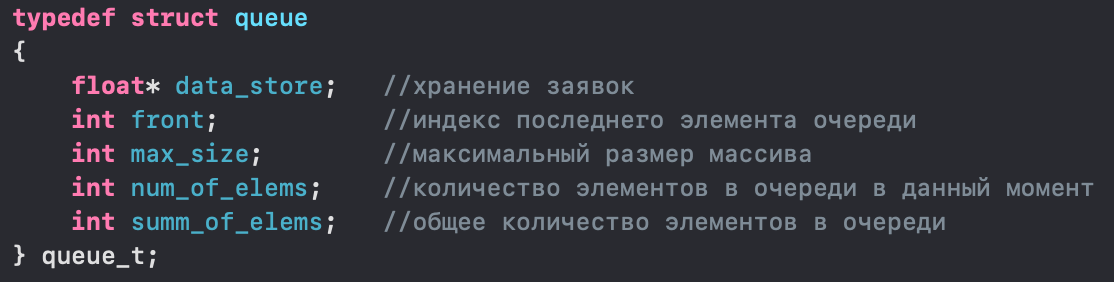


Результат работы программы:

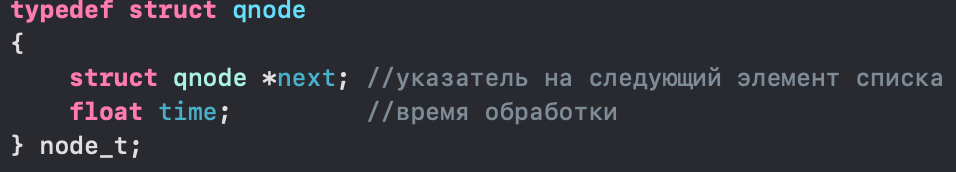


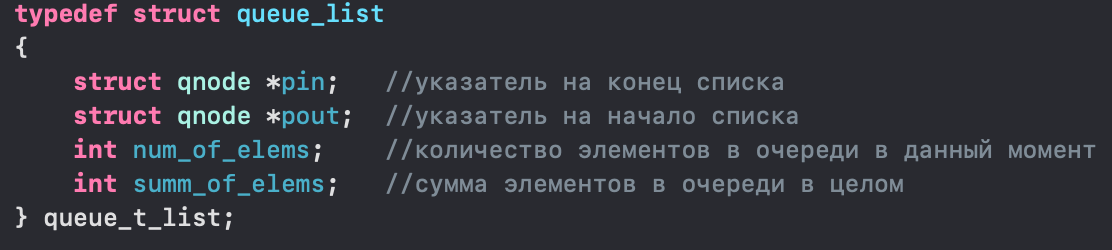
(Программа останавливает обработку заявок и не дает итогового результата при переполнении массива).

**Используемые структуры**

**При реализации массивом:**  
  
Структура для хранения очереди:  


**При реализации списком:**

Структура узла списка:  
  
****

Структура очереди:  
  
****

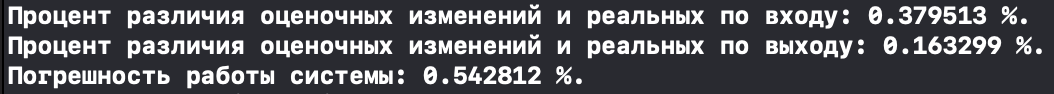
**Время моделирования**

Среднее время создания заявок: (6 + 0) / 2 = 3е.в.

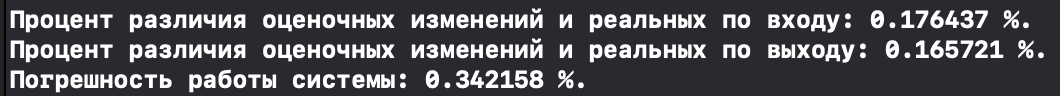
Среднее время обработки заявок (1 + 0) / 2 = 0.5е.в.

1000 заявок создается за 3 \* 1000 = 3000 е.в.

Данные на одном из экспериментов для списка:



Для массива:



Погрешность в целом примерно равна в обеих реализациях, колеблется от 0.2% до 2%. При работе программы возникает фрагментация памяти. Несмотря на это, освобожденные адреса чаще всего все-таки опять занимаются, так как программе нужно очень большое количество адресов.

При представлении очереди в виде списка используется большее количество памяти для хранения указателей. Также к недостаткам очереди-списка можно отнести возникновение фрагментации памяти. По времени список тоже менее эффективен, так как он должен освобождать и выделять память каждый раз, когда добавляется, или удаляется элемент, а это времезатратно.

**Ответы на вопросы**

1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с одной стороны, а исключение – с другой стороны.

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

В списке память под элемент очереди выделяется непосредственно в процессе его добавления. Объем памяти, который занимает очередь, изменяется в процессе выполнения программы и напрямую зависит от количества элементов в очереди в каждый момент времени. При реализации очереди массивом выделяется последовательная область памяти константного размера. Выделение памяти происходит в начале работы программы. При необходимости память перевыделяется.

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

В массиве освобождение памяти происходит в конце работы программы (или при удалении очереди). При удалении элемента из очереди происходит только смещение указателя. В списке при удалении элемента из очереди происходит освобождение памяти, которая была выделена под этот элемент.

1. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди хвостовой элемент из нее удаляется.

1. Каким образом эффективнее реализовывать очередь. От чего это зависит?

Если необходимо избежать фрагментации памяти, то лучше использовать очередь на массиве. Однако такой способ лучше использовать в том случае, если заранее известно количество элементов в очереди. Иначе лучше реализовывать список.

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

Очередь лучше реализовывать с помощью указателей, если новые элементы в среднем появляются реже, чем происходит полное очищение очереди – в общем случае фрагментация не возникает.

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

При реализации очереди массивом не возникает фрагментации памяти, однако может произойти переполнение очереди, а также затрачивается дополнительное время на сдвиг элементов (можно исключить сдвиг используя кольцевой массив). При реализации очереди списком затрачивается большее количество времени при добавлении нового элемента, для хранения указателей требуется дополнительная память.

1. Что такое фрагментация памяти?

При последовательных запросах на выделение и освобождении памяти под элемент не всегда выделяется память, которая была только что освобождена.

1. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на переполнение очереди, фрагментацию памяти при реализации очереди списком.

1. 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, однако указатель на этот блок может остаться в программе. Попытка считать данные из этого блока может привести к ошибке программы или неверному результату, поскольку они могут быть уже изменены.