

ФАКУЛЬТЕТ: Информатика и системы управления

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**«Обработка очередей»**

Студент: Зайцева Алена Андреевна

Группа ИУ7 – 32Б

*2020 г.*

Оглавление

[1. Описание условия задачи 2](#_Toc56551393)

[2. Описание ТЗ 4](#_Toc56551394)

[1) Описание исходных данных: 4](#_Toc56551395)

[2) Описание результатов моделирования: 4](#_Toc56551396)

[3) Описание задачи, реализуемой программой: 5](#_Toc56551397)

[4) Способ обращения к программе: 5](#_Toc56551398)

[5) Описание возможных аварийных ситуаций и ошибок пользователя. 5](#_Toc56551399)

[3. Описание внутренних структур данных. 6](#_Toc56551400)

[4. Описание алгоритма 6](#_Toc56551401)

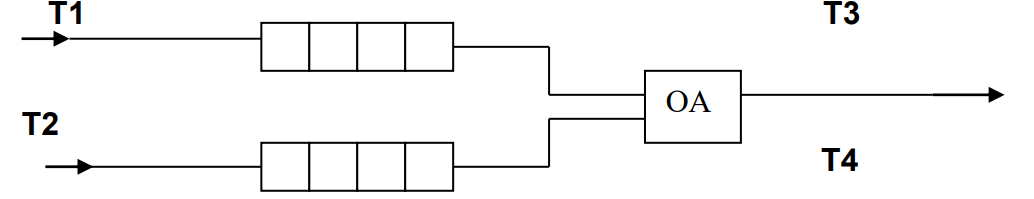
[5. Набор тестов с указанием, что проверяется 7](#_Toc56551402)

[6. Оценка эффективности 12](#_Toc56551403)

[7. Ответы на вопросы 14](#_Toc56551404)

# 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 **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 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 ждет первого освобождения ОА и далее поступает на обслуживание (система с относительным приоритетом).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа. Выдать на экран после обслуживания каждых 100 заявок 1-го типа информацию о текущей и средней длине каждой очереди, количестве вошедших и вышедших заявок и о среднем времени пребывания заявок в очереди.

В конце процесса выдать общее время моделирования, время простоя ОА, 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.

Очередь необходимо представить в виде вектора и списка. Необходимо сравнить эффективность различного представления очереди по времени выполнения программы и по требуемой памяти.

Рассогласование между средними ожидаемыми временами и временами, полученными в моделирующей программе должно быть не больше 2–3%.

(Примеры расчета ожидаемого времени – в разделе тестирование)

При разработке интерфейса программы следует предусмотреть:

* возможность изменения времен прихода и обработки заявок с экрана;
* наличие пояснений при выводе результата;
* вывод процента погрешности работы программы.
* Кроме того, нужно вывести на экран время выполнения программы при реализации очереди списком и массивом, а также требуемый при этом объем памяти

# Описание ТЗ

Внешняя спецификация:

## Описание исходных данных:

* 1. Номер команды: целое число в диапазоне от 0до 4.
* 0 – Выход из программы
* 1 – Вывод текущих параметров моделирования
* 2 – Изменение параметров моделирования
* 3 – Моделирование с помощью очередей – списков (при необходимости после моделирования будет предложено вывести информацию о работе с памятью)
* 4 – Моделирование с помощью очередей - массивов
  1. Данные, необходимые для изменения параметров моделирования
* Интервалы времени (вещественные, >= 0, min <= max)
* Количество элементов в первой очереди, которые надо обработать (целое, >0)
* Количество элементов в первой очереди, после обработки которых надо выводить промежуточные результаты (целое, >0, <= количества элементов в первой очереди, которые надо обработать)

## Описание результатов моделирования:

* 1. Через указанные промежутки будет выводиться информация о:
     1. текущей и средней длине каждой очереди,
     2. количестве вошедших и вышедших заявок в каждой очереди,
     3. о среднем времени пребывания заявок каждого типа в очереди.
  2. В конце моделирования выводится информация о:
     1. Общем время моделирования,
     2. Времени простоя ОА,
     3. Количестве вошедших в систему и вышедших из нее заявок обоих типов
     4. Ожидаемом времени моделирования (теоретическом, с указанием метода расчета)
     5. Проценте расхождения ожидаемого и полученного времени моделирования
     6. Объеме времени и памяти, необходимом при реализации очередей в данном виде
     7. При представлении очередей в виде списка – об особенностях выделения памяти

## Описание задачи, реализуемой программой:

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа. Выдать на экран после обслуживания каждых 100 заявок 1-го типа информацию о текущей и средней длине каждой очереди, количестве вошедших и вышедших заявок и о среднем времени пребывания заявок в очереди.

В конце процесса выдать 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.

## Способ обращения к программе:

Программа запускается из папки, содержащей исполняемый файл, через терминал.

## Описание возможных аварийных ситуаций и ошибок пользователя.

Возможные ошибки пользователя (для обеих реализаций стека):

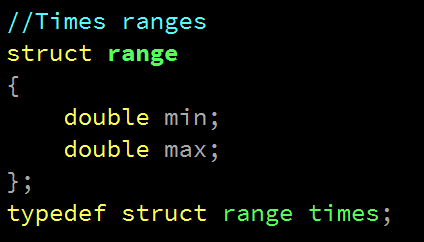
1. Ввод несуществующей команды: нецелочисленное значение или целое, меньшее 0 или большее 4
2. Неверные интервалы времени (min>max)
3. Нецелочисленное значения для количества заявок первого типа, которые необходимо обработать
4. Нецелочисленное значения для количества заявок первого типа, после обработки которых необходимо выводить промежуточные результаты, или это количество> общего количества заявок первого типа, которые необходимо обработать.
5. Одна из очередей может переполниться (максимальная вместимость каждой очереди – 10000 заявок) – программа выведет соответствующее сообщение и завершит моделирование

Возможные аварийные ситуации:

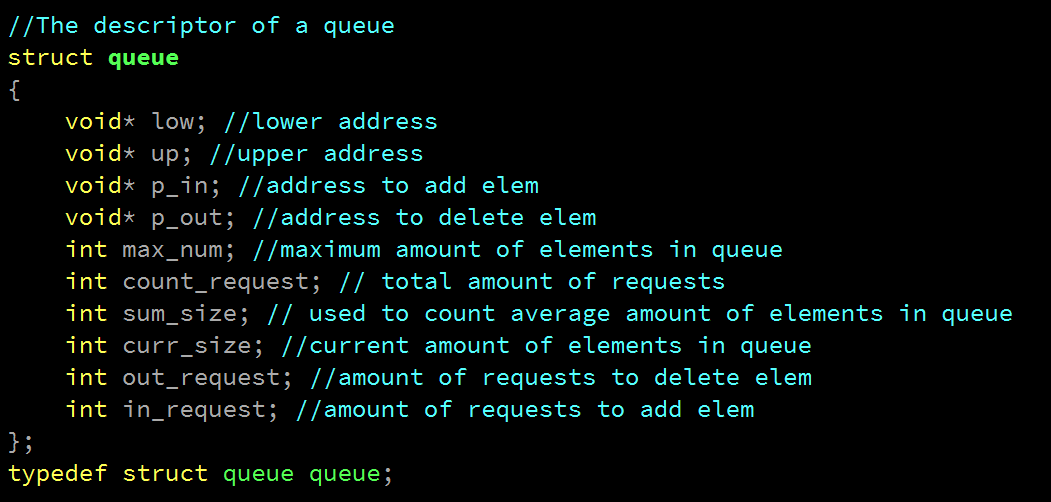
1. при вводе числового поля пользователь ничего не вводит – программа будет ждать ввода и не завершится, пока не встретит хоть один непробельный символ.
2. Компьютер не сможет выделить необходимое количество памяти – программа выведет сообщение об ошибке и завершится.

## Описание внутренних структур данных.

Интервалы времени:

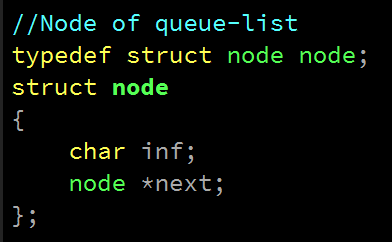


Дескриптор очереди:



Реализация очереди в виде списка:

Узел



И указатель на «голову»



Реализация очереди в виде массива:



## Описание алгоритма

1. Выводится общая информация о программе и список возможных команд.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять одну из команд:
3. Выход из программы
4. Вывод текущих параметров моделирования
5. Изменение параметров моделирования
6. Моделирование с помощью очередей – списков (при необходимости после моделирования будет предложено вывести информацию о работе с памятью)
7. Моделирование с помощью очередей - массивов

На каждом этапе контролируется успешность ввода и выделения памяти, корректность последовательности действий, пустота или переполнение стека. В случае ошибки программа выводит соответствующее сообщение.

При моделировании:

* + 1. Создается очередь в соответствующей реализации
    2. Проводится моделирование и вывод полученных результатов
    3. Выводятся ожидаемые (теоретические) результаты и процент расхождения с полученными
    4. Выводится информация об объеме времени и памяти, необходимом при реализации очередей в данном виде

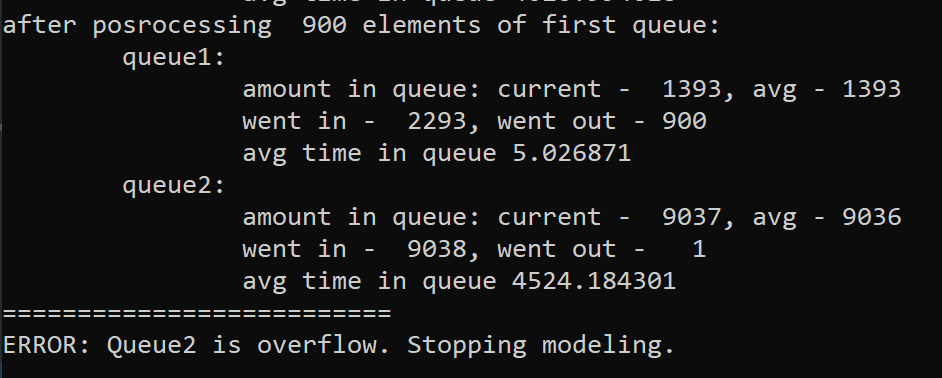
Код содержит комментарии по используемым в программе подпрограммам.

# Набор тестов с указанием, что проверяется

Тесты, проверяющие ввод:

|  |  |  |
| --- | --- | --- |
| Что проверяет | Ввод | Результат |
| Несуществующая команда | 5 или -1 или a | Сообщение об ошибке, вывод списка возможных команд и предложение вновь ввести команду |
| Некорректное количество элементов первой очереди, которые необходимо обработать | 0 или a | Сообщение об ошибке и предложение попробовать снова |
| Некорректное количество элементов, после которых нужно выводить промежуточную информацию | 0 или a или (больше максимального количества элементов) | Сообщение об ошибке и предложение попробовать снова |
| Некорректный интервал времени | (f 1) или (-1 10)  или (10 1) | Сообщение об ошибке и предложение попробовать снова |

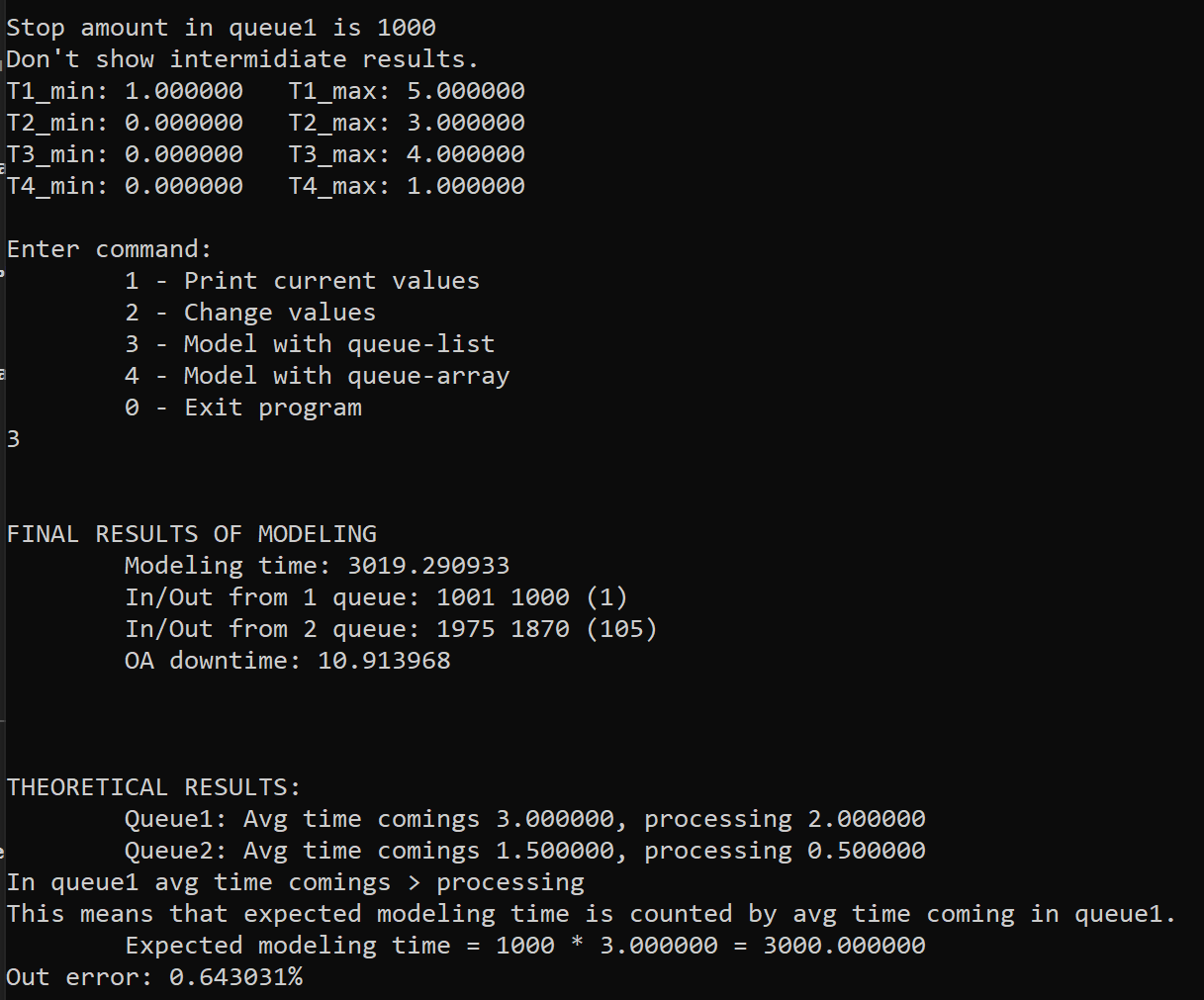
Проверка переполнения очереди-массива:



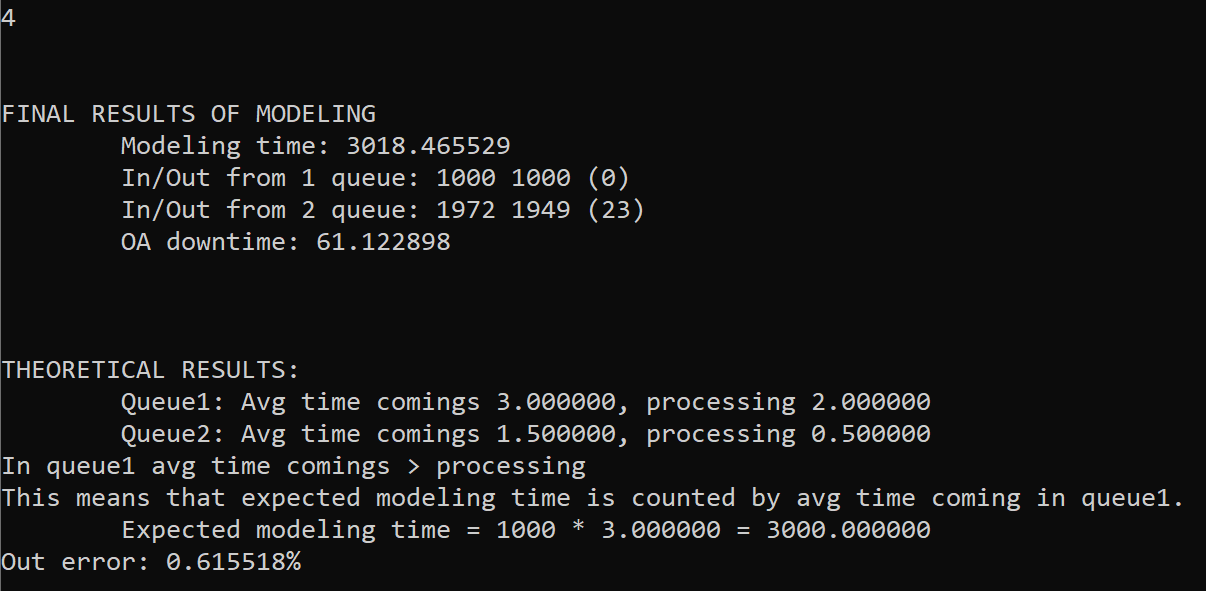
Тесты, проверяющие моделирование:

Время моделирования определяется временем прихода заявки (avg(T1) > avg(T3))

Очередь - список

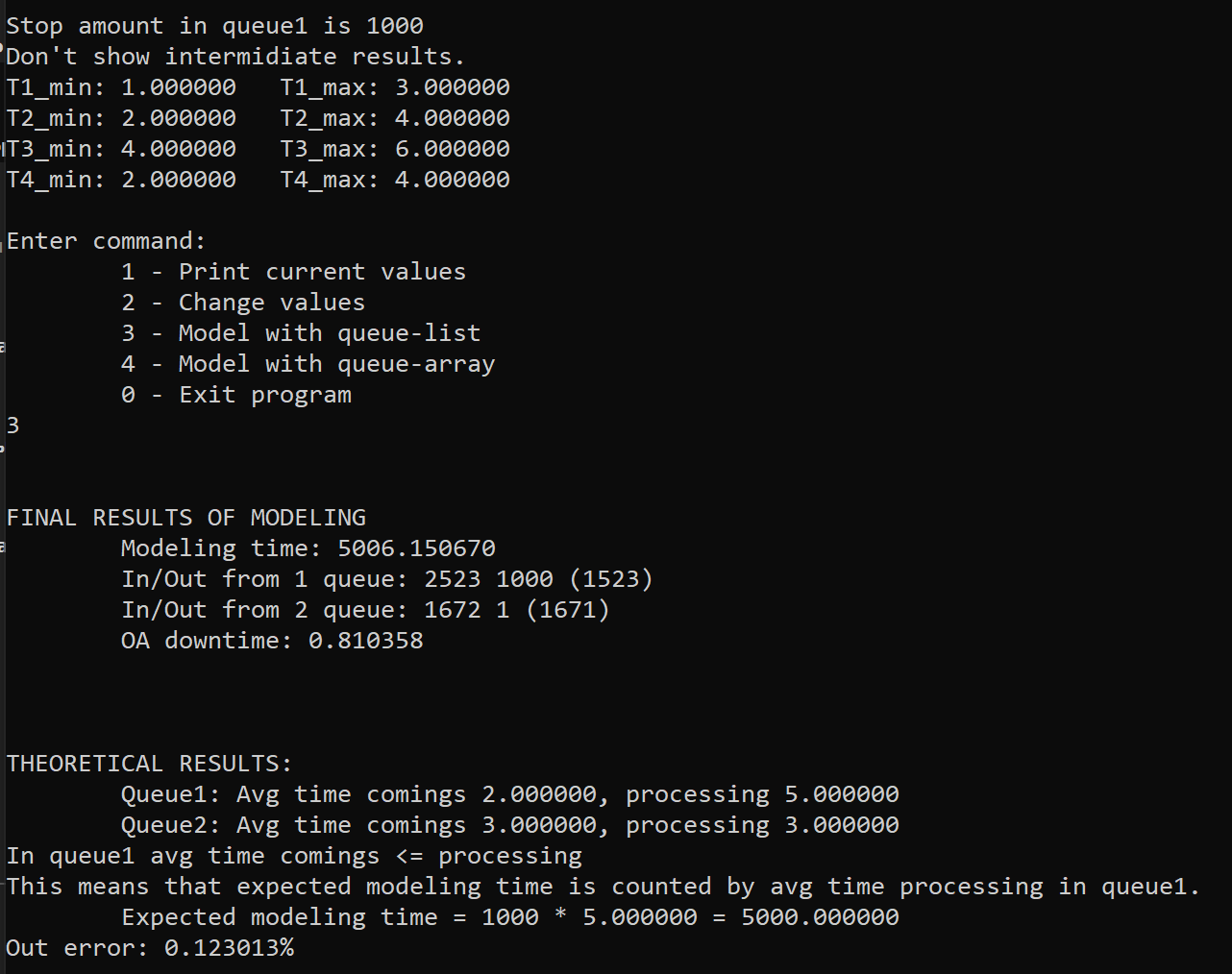


Очередь - массив

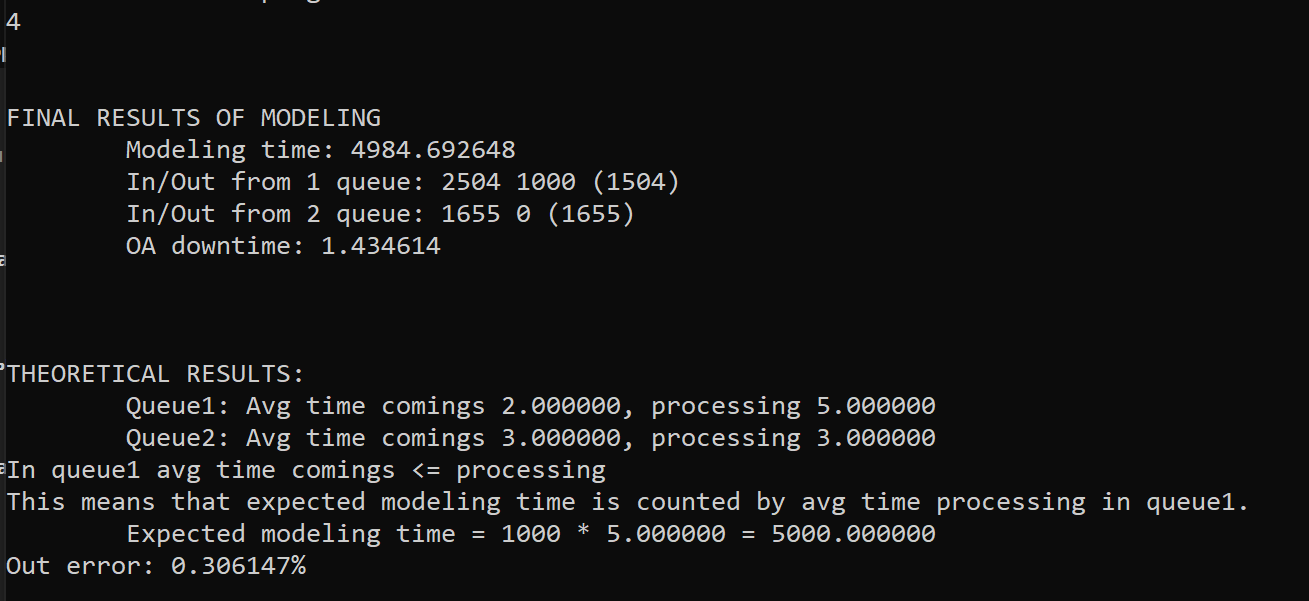


Время моделирования определяется временем обработки заявки (avg(T1) <= avg(T3))

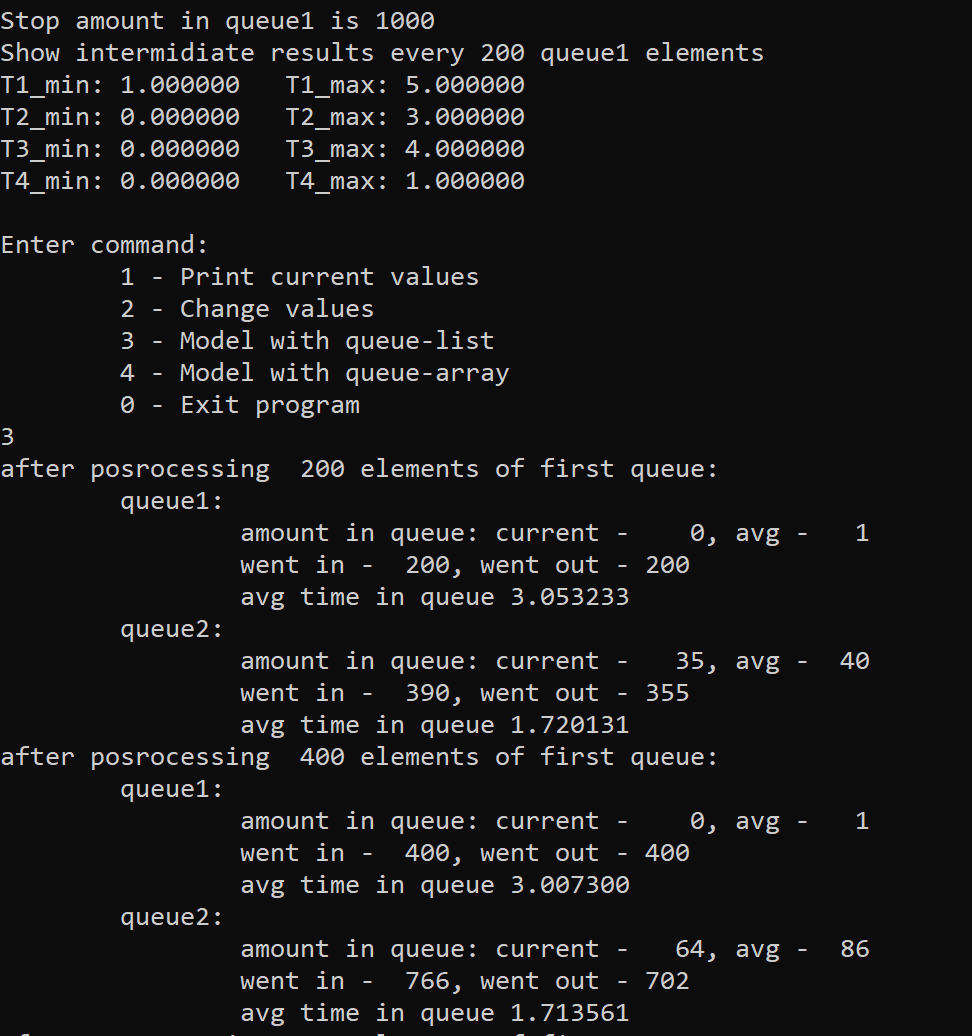
Очередь – список

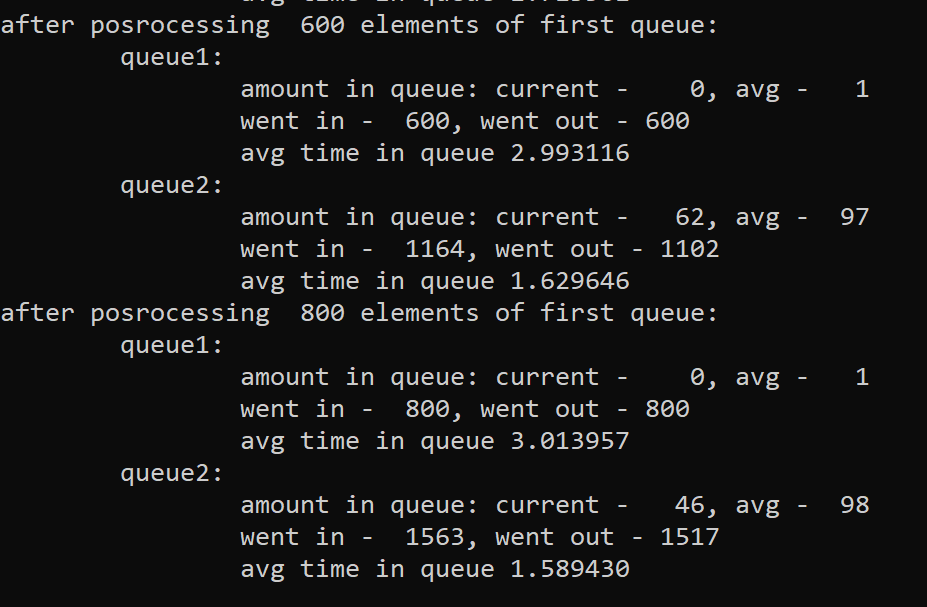


Очередь – массив

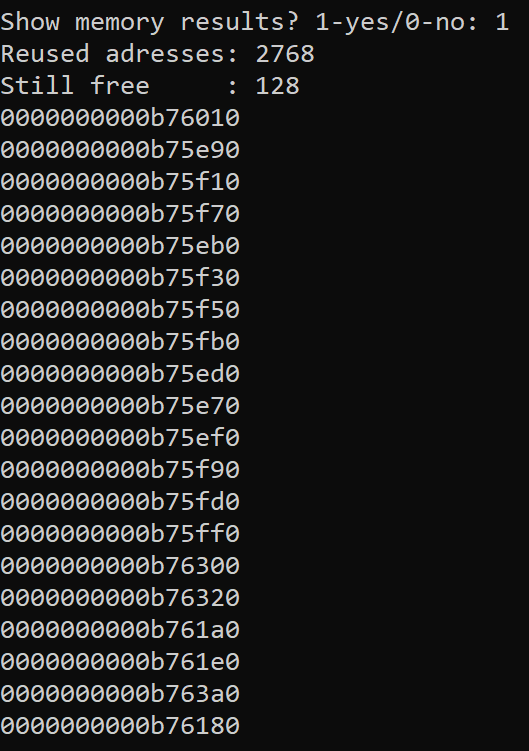


Проверка расчета промежуточных результатов





Информация об использованной памяти:

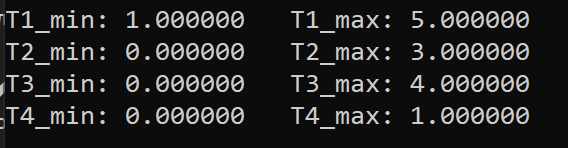


Большая часть адресов используются повторно.

# Оценка эффективности

Оценим эффективность различных реализаций очереди (в виде массива и в виде списка) при данном моделировании на различном количестве элементов в первой очереди, которые необходимо обработать (10, 100, 1000, 5000)

Остальные параметры моделирования– как в задании:



*Эффективность по памяти*

При реализации очередей в виде массивов, необходимо сразу хранить:

* Два указателя на каждую из очередей (2 \* sizeof(char \*) = 2\*8 = 16)
* Два массива элементов максимально допустимой длины (2 \* sizeof(char) \* 10000 = 2\*1\*2000 = 20000

Итого – 20016 байт, независимо от того, сколько элементов первой очереди необходимо обработать.

При реализации очередей в виде списков, необходимо сразу хранить:

* Два указателя на каждую из очередей (2 \* sizeof(char \*) = 16)

А также подсчитывается среднее количество узлов, которые во время выполнения хранятся в программе. Они занимают sizeof(node)\*avg\_nodes = (sizeof(char) + sizeof(node \*)) \* avg\_nodes = 16 (из-за выравнивания) \* avg\_nodes.

Результаты:

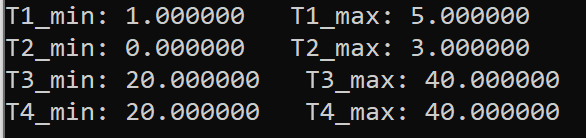
**Объем занимаемой памяти (в байтах)**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Avg\_nodes | Очереди - массивы | Очереди - списки |
| 10 | 17 | 20016 | 288 |
| 100 | 25 | 20016 | 416 |
| 1000 | 132 | 20016 | 2128 |
| 5000 | 174 | 20016 | 2800 |

Итак, при данных параметрах моделирования реализация очереди в виде списка сильно выигрывает у массива по памяти.

Но с ростом количества элементов в первой очереди, которые необходимо обработать, разрыв уменьшается, так как элементы в очередях все сильнее накапливаются.

При этом нужно учитывать, что эффективность по памяти будет зависеть от параметров моделирования: если время обработки ОА будет большим, то в очередях придется хранить много элементов, и тогда очередь – список будет проигрывать массиву. Например, при необходимости обработать 100 элементов первой очереди и таких параметрах:



Очереди – массивы все так же будут занимать 20016 байт, а вот очередь – список, из-за необходимости хранить avg\_nodes = 1976034, займет уже 31616560 байт

Таким образом, при выборе способа хранения очередей в зависимости от эффективности по памяти, нужно учитывать, насколько сильно буду накапливаться заявки в очередях.

*Эффективность по времени*

Время добавления элемента в очередь для обоих реализаций – константное, так как массив – структура данных с константным временем доступа к элементу, а в список элемент добавляется в “голову”, то есть нет необходимости в проходе по всем элементам списка.

Но время удаления элемента из очереди для массива по той же причине - константное, а из списка элемент удаляется из “хвоста”, то есть появляется необходимости в проходе по всем элементам списка, чтобы добраться до последнего.

При этом для добавлении удалении элемента при реализации очереди в виде списка необходимо выделять и освобождать память, что серьезно замедляет процесс обработки.

Поэтому отношение времени моделирования при представлении массивом к времени моделирования при представлении списком должно уменьшаться с ростом числа элементов.

**Время моделирования (в тактах процессора, среднее из 5 измерений)**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Очереди - массивы | Очереди - списки | Отношение массив/список |
| 10 | 13.989 | 38.283 | 0.37 |
| 100 | 110.239 | 354.144 | 0.31 |
| 1000 | 1.089.968 | 3.690.834 | 0.29 |
| 5000 | 4.877.208 | 30.856.390 | 0.16 |

Таким образом, представление очередей в виде массивов эффективней по времени, чем в виде списков, причем чем большее количество элементов нужно обработать, тем заметнее выигрыш очередей-массивов.

# Ответы на вопросы

* + - 1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с одной стороны (с «хвоста»), а исключение – с другой стороны (с «головы»). Принцип работы очереди: первым пришел – первым вышел, т. е. First In – First Out (FIFO).

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

При моделировании очереди на основе одномерного массива выделяется последовательная область памяти из m мест по L байт, где L – размер поля данных для одного элемента размещаемого типа. Память может выделяться как статически, так и динамически, в зависимости от решения программиста.

При реализации очереди на основе односвязного линейного списка, каждый элемент которого содержит информационное поле и поле с указателем «вперед» в статической памяти можно либо хранить адрес начала и конца очереди, либо – адрес начала очереди и количество элементов. Память под сами элементы выделяются динамически – при добавлении очередного элемента.

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

В очереди на основе массива исключается элемент, находящийся по адресу (индексу) Pout, а указатель Pout перемещается к следующему элементу (то есть фактически память не освобождается при удалении очередного элемента).

В очереди на основе списка также исключается элемент, находящийся по адресу Pout. Указатель Pout при этом переместится на предыдущий элемент, а память из-под исключенного элемента необходимо освободить.

* + - 1. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди текущий элемент из нее удаляется.

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

Прежде всего необходимо выбрать, что важнее - время или память.

Если важнее память, то нужно продумать, насколько сильно будет накапливаться очередь: если она будет всегда практически полностью заполнена, то эффективнее массив, в остальных ситуациях – список. При реализации очереди в виде списка списком легче добавить и удалить элемент, но при этом может возникнуть фрагментация памяти.

Если важнее время, то эффективнее использовать массив.

Но при реализации массивом и сильной заполненности при удалении необходимо сдвигать все его элементы, что при больших размерах очереди затратно по времени. Поэтому здесь тоже стоит продумать, насколько сильно будет накапливаться очередь.

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

Если очередь не будет сильно накапливаться: по памяти – список, по времени – массив.

Если очередь будет сильно накапливаться: по памяти – массив, по времени – зависит от того, будем ли мы сдвигать элементы очереди при добавлении/удалении.

* + - 1. 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. При реализации списком понадобится меньше памяти, но может возникнуть ее фрагментация, а при удалении очередного элемента придется пройти по всем элементам очереди, чтобы добраться до конца.

* + - 1. Что такое фрагментация памяти?

Фрагментация– это дробление памяти на мелкие несмежные свободные области маленького размера, то есть чередование занятых и свободных участков памяти при последовательных запросах на добавление и удаление. Свободные участки могут быть слишком малы, чтобы хранить в них нужную информацию.

* + - 1. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо проверить правильность работы программы при различном заполнении очередей, отследить переполнение очереди, если очередь в программе ограничена, при реализации очереди списком необходимо тщательно следить за освобождением памяти при удалении элемента из очереди.

* + - 1. 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.