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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
«ОБРАБОТКА ОЧЕРЕДЕЙ»

Студент: Лысцев Никита Дмитриевич

Группа: ИУ7-33Б

Студент _____ Лысцев Н.Д

подпись, дата *фамилия,и.о.*

Преподаватель _____ Барышникова М. Ю.

подпись, дата *фамилия,и.о.*

Оценка _____

2022 г

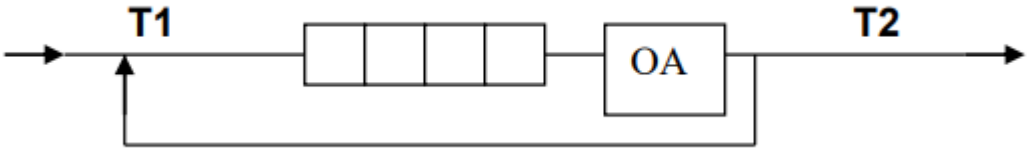
Оглавление

1. Цель работы

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.

2. Описание условия задачи

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



Заявки поступают в "хвост" очереди по случайному закону с интервалом времени T1, 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 T2 от 0 до 1 е.в., Каждая заявка после ОА вновь поступает в "хвост" очереди, совершая всего 5 циклов обслуживания, после чего покидает систему (все времена – вещественного типа). В начале процесса в системе заявок нет.

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок, выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди, а в конце процесса - общее время моделирования и количестве вошедших в систему и вышедших из нее заявок, количестве срабатываний ОА, время простоя аппарата. По

требованию пользователя выдать на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

3. Описание технического задания

3.1. Входные данные

Целое число от 0 до 1 – выбор пользователя – решать задачу моделирования СМО (число 0) или работать с очередью вручную (число 1).

В случае, если пользователь выбрал решение задачи моделирования СМО, то:

Целое число от 0 до 4 – номер пункта меню, позволяющий пользователю осуществить смену времен добавления и обработки заявок, а также осуществить моделирование СМО.

В случае, если пользователь выбрал работу с очередью вручную, то:

Целое число от 0 до 13 – номер пункта меню, позволяющий пользователю взаимодействовать с очередью (добавлять элемент, удалять элемент, очищать очередь, создавать очередь) с реализацией очереди с помощью массива и с реализацией очереди с помощью односвязного линейного списка.

Целое число – размерность очереди для его реализации с помощью массива (не более 10000 элементов).

Элемент очереди – целое число.

3.2. Выходные данные

При выводе каждых 100 вышедших заявок – результаты моделирования.

При выходе 1000 заявок – окончательные результаты моделирования.

Вывод использованных адресов при реализации очереди списком.

Данные об эффективности.

Выход из программы

3.3. Описание задачи, реализуемой программой

Задача программы – моделирование СМО, используя такую структуру данных, как очередь.

3.4. Способ обращения к программе

Ввод и вывод всех данных осуществляется через консоль.

Программа собирается с помощью специального make-файла, и может быть запущена в командной строке с помощью слова make.

3.5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Пустое поле ввода – ничего не произойдет, программа будет ждать ввода пользователя;
- Переполнение очереди при моделировании – программа выдаст сообщение о переполнении, процесс моделирования прервется программа и завершит свою работу с ненулевым кодом возврата.

Ошибки пользователя:

- Некорректный ввод: ввод неверного пункта меню, ввод неверной размерности для создания стека как массива, добавление в переполненную очередь, удаление из пустой очереди, ввод не валидного элемента очереди.

4. Описание внутренних структур данных

Элемент очереди -- заявка -- для реализации очереди как массивом, так и односвязным списком описывается структурой типа «elem_t».

```
typedef struct elem_t elem_t;

/// @brief заявка
struct elem_t
{
    double add_time;     // время добавления
    double process_time; // время обработки
    int count_iter;      // количество проходов
};
```

Листинг 1. Структура «elem_t», содержащая описание одной заявки

Поля структуры «elem_t»:

1. add_time – вещественное число – время добавления заявки;
2. process_time – вещественное число – время обработки заявки;
3. count_iter – целое число, количество проходов через обслуживающий аппарат.

Реализация очереди с помощью одномерного динамического массива описывается структурой «queue_array_t».

```
typedef struct queue_array_t queue_array_t;

/// @brief очередь как массив
struct queue_array_t
{
    elem_t *queue_array; // массив заявок
    int p_in;            // индекс очередного добавленного элемента
    int p_out;           // индекс очередного исключающего элемента
    int size;            // размер массива заявок
};
```

Листинг 2. Реализация стека как массива с помощью структуры типа «stack_array_t»:

Поля структуры «queue_array_t»:

1. queue_array – массив структур типа «elem_t», содержащий в себе элементы очереди;

2. p_in -- целочисленная переменная, содержащая в себе индекс хвоста очереди, то есть индекс последнего добавленного элемента;
3. p_out – целочисленная переменная, содержащая в себе индекс очередного исключающего элемента очереди;
4. p_out – целочисленная переменная, содержащая в себе максимальный размер очереди.

Реализация очереди с помощью линейного односвязного списка описывается структурой типа «queue_list_t»

```
typedef struct list_t list_t;
typedef struct queue_list_t queue_list_t;

/// @brief односвязный список, содержащий данные
struct list_t
{
    elem_t data;  // данные
    list_t *next; // указатель на следующий элемент
};

/// @brief очередь в виде односвязного списка
struct queue_list_t
{
    list_t *queue_list; // очередь в виде списка
    int size;           // количество элементов в списке
};
```

Листинг 3. Реализация стека как линейного односвязного списка с помощью структуры типа «stack_list_t»

Поля структуры «queue_list_t»:

1. queue_list – переменная структурного типа «list_t» -- односвязный линейный список, содержащий в себе элементы очереди;
2. size – количество элементов очереди.

Элемент очереди (узел) односвязного линейного списка представлен структурой «list_t»

Поля структуры «list_t»

1. data – переменная структурного типа «elem_t», содержащая информацию о заявке;

2. next – указатель на следующий элемент очереди.

Также при реализации стека линейным односвязным списком для вывода ранее освобожденных адресов в результате удаления элемента из стека или очищения стека используется структура типа «free_addr_t».

```
typedef struct addr_t addr_t;
typedef struct free_addr_t free_addr_t;

/// @brief адрес
struct addr_t
{
    size_t *addr;
    bool check_free;
    bool check_create;
};

/// @brief массив свободных адресов
struct free_addr_t
{
    // массив ранее высвобожденных адресов
    addr_t *free_addrs;
    int top;  // индекс последнего  добавленного элемента
    int size; // размер массива адресов
};
```

Листинг 4. Структура типа «free_addr_t»

Поля структуры «free_addr_t»:

1. free_addrs – массив указателей, содержащий ранее освобожденные адреса для реализации очереди односвязным списком. Длина этого массива не превышает 100000 элементов;
2. top – индекс последнего добавленного элемента в массив адресов.
3. size – размер массива адресов.

Для хранения одного адреса используется структура «addr_t»

Поля структуры «addr_t»:

4. addr – значение адреса;
5. check_free – значение для освобождения адреса;
6. check_create – значение для создания адреса.

Для хранения параметров моделирования используется структура «param_t»

```
typedef struct param_t param_t;

/// @brief Структура параметров моделирования
struct param_t
{
    double min_add_time;     // минимальное время добавления
    double max_add_time;     // максимальное время добавления
    double min_process_time; // минимальное время обработки
    double max_process_time; // максимальное время обработки
    int count_appl;          // количество обрабатываемых заявок
    int count_print_appl;    // диапазон заявок для вывода
};
```

Листинг 5. Структура параметров моделирования «param_t»

Поля структуры «param_t»:

1. min_add_time – минимальное время добавления заявки;
2. max_add_time – максимальное время добавления заявки;
3. min_process_time – минимальное время обработки заявки;
4. max_process_time – максимальное время обработки заявки;
5. count_appl – количество обрабатываемых заявок;
6. count_print_appl – диапазон заявок для вывода.

Для хранения данных моделирования используется структура «model_t»

```
typedef struct model_t model_t;

/// @brief Структура, содержащая данные моделирования
struct model_t
{
    int curr_len_queue;      // текущая длина очереди
    double av_len_queue;     // средняя длина очереди
    double model_time;       // общее время моделирования
    double input_time;       // общее время добавления
    double output_time;      // общее время обработки
    double downtime;         // время простоя
    int count_input;         // количество вошедших заявок
    int count_output;        // количество вышедних заявок
    int count_calls_machine; // количество вызовов аппарата
    long double work_time;   // время работы (в микросекундах)
    int memory_size;         // затраченный обьем памяти
};
```

Листинг 6. Структура данных моделирования «model_t»

Поля структуры «model_t»:

1. curr_len_queue – текущая длина очереди;
2. av_len_queue – средняя длина очереди;
3. model_time – общее время моделирования;
4. input_time – общее время добавления заявок;
5. output_time – общее время обработки заявок;
6. downtime – время простоя обслуживающего аппарата;
7. count_input – количество вошедших заявок;
8. count_output – количество вышедших заявок;
9. count_calls_machine – количество срабатываний аппарата;
10. work_time – время работы моделирования (в микросекундах)
11. memory_size – требуемый для моделирования максимальный объем памяти в байтах.

4.1. Описание алгоритма

Взаимодействие пользователя с программой осуществляется через консоль с помощью специального меню, в котором пользователю предлагается выполнить то, или иной действие.

В начале меню в одном предложении кратко изложена суть программы. Далее пользователю предлагается выбрать: решить задачу моделирования СМО, либо поработать с очередью вручную.

```
Программа для работы с очередью.
Позволяет смоделировать СМО, а также работать с очередью вручную

Введите значение 0, если хотите решить задачу моделирования СМО, или введите значение 1, если хотите
работать с очередью в ручном режиме:
```

Листинг 7. Фрагмент меню взаимодействия

Если пользователь выбирает 0, то следующим шагом пользователю предлагается выбрать один из пунктов меню, представленного ниже

```
--------------------------------------------------------|
Параметры обработки и добавления по умолчанию:
Добавление: от 0 до 6 е.в.
Обработка:  от 0 до 1 е.в.
```

```
Текущие параметры обработки и добавления:
Добавление: от   0.000000 до   6.000000 е.в.
Обработка:  от   0.000000 до   1.000000 е.в.
1 - Изменить время обработки;
2 - Изменить время добавления;
3 - Смоделировать СМО, используя очередь в виде списка;
4 - Смоделировать СМО, используя очередь в виде массива;
5 - Вывести информацию об использованных и освобожденных
    адресах при моделировании очереди в виде списка.

0 - Выйти из программы.
-------------------------------------------------------|

Выберите пункт меню:
```

Листинг 8. Меню взаимодействия для моделирования

Если пользователь выбирает 1, то следующим шагом пользователю предлагается выбрать один из пунктов меню, представленного ниже

```
-------------------------------------------------------|
Очередь реализована двумя способами:
а) массивом;
б) списком.

Операции с очередью в виде массива:
1  - создать очередь;
2  - добавить элемент в очередь;
3  - удалить элемент из очереди;
4  - вывести текущее состояние очереди;
5  - очистить очередь;

Операции с очередью в виде односвязного линейного списка:
6  - создать очередь;
7  - добавить элемент в очередь;
8  - удалить элемент из очереди;
9  - вывести текущее состояние очереди;
10 - очистить очередь;
11 - вывести массив ранее освобожденных адресов;
12 - вывести сравнение времен работы различных функции
     для работы с очередью
13 - вывести сравнение представления по памяти
     очереди как списка и очереди как массива

0  - выйти из программы.

-------------------------------------------------------|
Выберите пункт меню:
```

Листинг 9. Меню взаимодействия для ручной работы с очередью

4.2. Ограничения на входные данные

Максимальная вместимость очереди в случае обоих реализаций – 100000 элементов.

Максимальная вместимость массива свободных адресов – 100000 элементов.

Элементом очереди является заявка. При ручной работе с очередью элементом очереди будет одно целое число.

Попытка удаления из пустой очереди или добавление в полностью заполненную очередь обрабатывается программой, выдавая соответствующие сообщения, но не завершая программу с ненулевым кодом возврата.

При вводе любых не валидных данных программа выдает соответствующее сообщение и завершает работу с ненулевым кодом возврата.

5. Оценка эффективности работы алгоритмов

Время работы функций для работы с очередью (микросекунды)

Количество элементов	**Добавление элемента**	
	Список	Массив
10	2.000000	2.000000
100	3.000000	2.000000
300	5.000000	2.000000
500	6.000000	2.000000
800	7.000000	1.000000
1000	7.000000	1.000000

Время работы функций для работы с очередью (микросекунды)

Количество элементов	**Удаление элемента**	
	Список	Массив
10	3.000000	3.000000
100	2.000000	10.000000
300	3.000000	54.000000
500	3.000000	49.000000
800	4.000000	143.000000
1000	7.000000	111.000000

Время работы функций для работы с очередью (микросекунды)

Количество элементов	**Очистка очереди**	
	Список	Массив

10	9.000000	4.000000
100	82.000000	4.000000
300	294.000000	10.000000
500	337.000000	13.000000
800	650.000000	30.000000
1000	630.000000	21.000000

Время работы функций для работы с очередью (микросекунды)

Создание очереди	
Список	**Массив**
2.000000	42.0000000

Размер памяти (в байтах), необходимый для хранения очереди

Количество элементов в очереди	**Массив**	**Список**
100	24036	3212
200	24036	6412
300	24036	9612
700	24036	25612
800	24036	28812
1000	24036	32012

6. Выводы

Операция добавления элемента в очередь при реализации очереди массивом выполняется быстрее, чем при реализации очереди списком, поскольку, используя массив, мы можем пользоваться операцией индексации, а при использовании списка должны от головы идти к хвосту.

Операция удаления элемента из очереди при реализации очереди списком выполняется быстрее (при небольшом количестве элементов (~100) – в 2-5 раз быстрее; при большом количестве элементов (~1000) – в 15-18 раз быстрее), чем при реализации очереди массивом, поскольку, используя список, мы просто удаляем данные из головы очереди, переопределяя голову, а удаление элемента в массиве сопровождается сдвигом к началу массива всех элементов после первого.

При небольшом количестве элементов очереди (~10) операция очистки очереди как списка происходит в два раза медленнее, чем при реализации очереди массивом. При большом количестве элементов очереди (~1000) операция очистки очереди как массива происходит в 20-30 раз быстрее, чем при реализации очереди списком.

Операция создания очереди как списка выполняется примерно в 20 раз быстрее, чем операция создания очереди как массива.

По результатам замеров можно сказать, что метод очереди в качестве списка, используется в том случае, когда размер очереди невелик, достигает примерно 30 элементов (~30%). Если в очереди будет много элементов, то предпочтительнее будет реализация очереди в виде массива.

Можно сказать, что эффективнее реализовывать очередь с помощью массива, поскольку большинство операций и очередью для этой реализации работают быстрее.

7. Расчет времени работы очереди

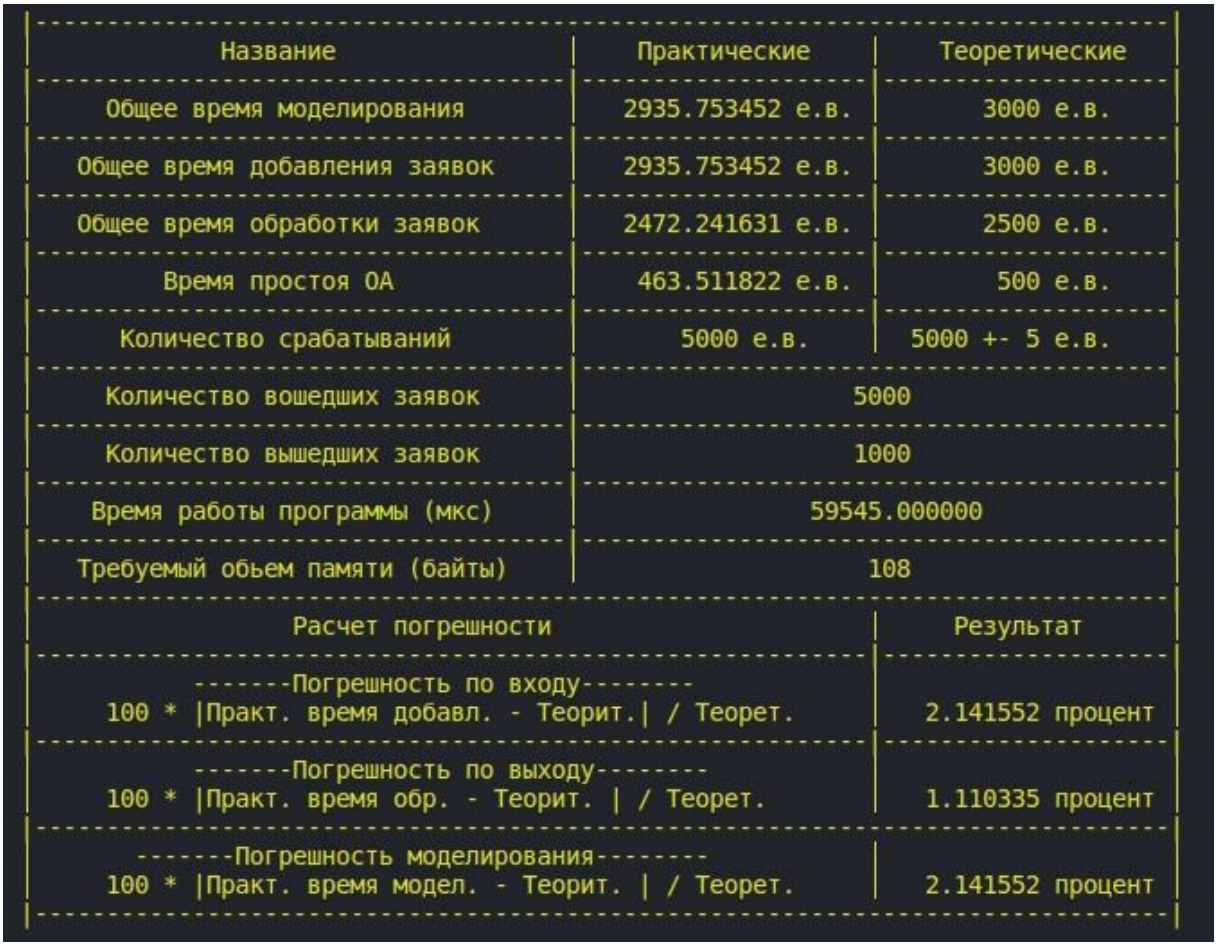
Минимальное время добавления T1 = 0 е.в. Минимальное время добавления T1 = 6 е.в. Минимальное время обработки T2 = 0 е.в. Максимальное время обработки T2 = 1 е.в.

Время обработки заявки лежит в интервале от 0 до 1 е.в., значит, среднее значение времени обработки одной заявки 0.5 е.в. Так как каждая заявка обрабатывается 5 раз до выхода из системы, то среднее время обработки одной заявки будет: 0.5*5=2.5 е.в., а общее время обработки 1000 заявок будет: 1000*2.5 = 2500 е.в. Для прихода 1000 заявок, если каждая из них приходит в среднем за 3 е.в., потребуется 3000 е.в. Следовательно, ОА работает с простоем и время моделирования будет определяться временем прихода заявок, т.е. оно должно быть равно 3000 е.в. Время простоя будет

равно разнице между временем обработки заявок и временем их обслуживания: 3000 – 2500 = 500 е.в.

На практике были получены следующие результаты:

Моделирование СМО с помощью очереди как списка:

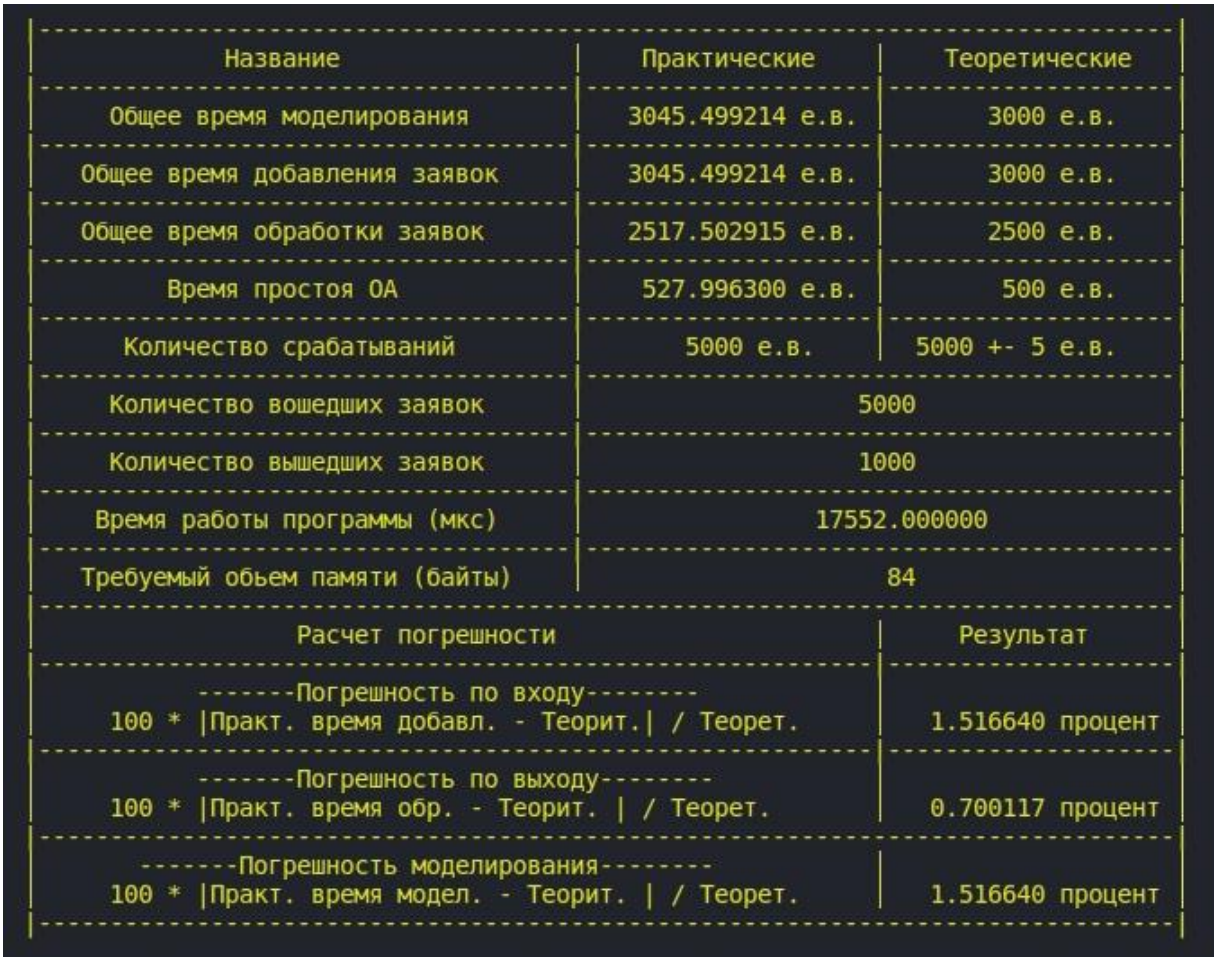


Название	Практические	Теоретические
Общее время моделирования	2935.753452 е.в.	3000 е.в.
Общее время добавления заявок	2935.753452 е.в.	3000 е.в.
Общее время обработки заявок	2472.241631 е.в.	2500 е.в.
Время простоя ОА	463.511822 е.в.	500 е.в.
Количество срабатываний	5000 е.в.	5000 +- 5 е.в.
Количество вошедших заявок	5000	
Количество вышедших заявок	1000	
Время работы программы (мкс)	59545.000000	
Требуемый обьем памяти (байты)	108	

Расчет погрешности	Результат
-------Погрешность по входу-------- 100 * |Практ. время добавл. - Теорит.| / Теорет.	2.141552 процент
-------Погрешность по выходу-------- 100 * |Практ. время обр. - Теорит. | / Теорет.	1.110335 процент
-------Погрешность моделирования-------- 100 * |Практ. время модел. - Теорит. | / Теорет.	2.141552 процент

Из результатов видно, что погрешность в пределах допустимой по заданию.

Моделирование СМО с помощью очереди как массива:



Название	Практические	Теоретические
Общее время моделирования	3045.499214 е.в.	3000 е.в.
Общее время добавления заявок	3045.499214 е.в.	3000 е.в.
Общее время обработки заявок	2517.502915 е.в.	2500 е.в.
Время простоя ОА	527.996300 е.в.	500 е.в.
Количество срабатываний	5000 е.в.	5000 +- 5 е.в.
Количество вошедших заявок	5000	
Количество вышедших заявок	1000	
Время работы программы (мкс)	17552.000000	
Требуемый обьем памяти (байты)	84	

Расчет погрешности	Результат
-------Погрешность по входу-------- 100 * |Практ. время добавл. - Теорит.| / Теорет.	1.516640 процент
-------Погрешность по выходу-------- 100 * |Практ. время обр. - Теорит. | / Теорет.	0.700117 процент
-------Погрешность моделирования-------- 100 * |Практ. время модел. - Теорит. | / Теорет.	1.516640 процент

Из результатов видно, что погрешность в пределах допустимой по заданию.

Анализируя время работы программы и требуемый объем памяти для очереди при каждой реализации можно увидеть, что моделирование СМО при помощи массива примерно в 3.5 раза эффективнее по времени, а также почти в 1.5 раза эффективнее по памяти.

8. Набор тестов

Позитивные тесты

№ Теста	Входные данные	Выходные данные	Результат
01	Начало работы, выбор пользователя равен 1	Вывод дополнительного меню для работы с очередью в ручном режиме	Ожидание следующего действия.
02	Начало работы, выбор пользователя равен 0	Вывод дополнительного меню для моделирования СМО	Ожидание следующего действия.
03	Выбор пользователя равен 0 Пункт меню 1 Ввод валидных значений времен обработки	Отображение измененных значений времен обработки в меню	Ожидание следующего действия.
04	Выбор пользователя равен 0 Пункт меню 2 Ввод валидных значений времен добавления	Отображение измененных значений времен добавления в меню	Ожидание следующего действия.
05	Выбор пользователя равен 0 Пункт меню 3 или 4	Вывод данных моделирования	Ожидание следующего действия.
06	Выбор пользователя равен 1 Пункт меню 1 Ввод валидных значений размерности очереди	Сообщение об успешном создании очереди как массива	Ожидание следующего действия.
07	Выбор пользователя равен 1 Пункт меню 2 Очередь создана ранее Ввод валидного элемента очереди	Сообщение об успешном добавлении элемента в очередь	Ожидание следующего действия.
08	Выбор пользователя равен 1 Пункт меню 2 Очередь не создана ранее	Сообщение о том, что очередь еще не создана	Ожидание следующего действия.
09	Выбор пользователя равен 1 Пункт меню 3 Очередь создана ранее и не пустая	Сообщение об успешном удалении элемента очереди	Ожидание следующего действия.
10	Выбор пользователя равен 1 Пункт меню 3 Очередь не создана ранее или пустая	Сообщение о том, что очередь либо не создана, либо пустая	Ожидание следующего действия.

10	Выбор пользователя равен 1 Пункт меню 4 Очередь создана ранее и не пустая	Вывод очереди на экран	Ожидание следующего действия.
11	Выбор пользователя равен 1 Пункт меню 4 Очередь не создана ранее или пустая	Сообщение о том, что очередь либо не создана, либо пустая	Ожидание следующего действия.
12	Выбор пользователя равен 1 Пункт меню 5 Очередь создана ранее	Сообщение об успешном очищении очереди	Ожидание следующего действия.
13	Выбор пользователя равен 1 Пункт меню 6 Очередь не создана ранее	Сообщение об успешном создании очереди как списка	Ожидание следующего действия.
14	Выбор пользователя равен 1 Пункт меню 6 Очередь создана ранее	Сообщение о том, что очередь уже была создана	Ожидание следующего действия.
15	Выбор пользователя равен 1 Пункт меню 7 Очередь создана ранее Ввод валидного элемента очереди	Сообщение об успешном добавлении элемента в очередь	Ожидание следующего действия.
16	Выбор пользователя равен 1 Пункт меню 7 Очередь не создана ранее	Сообщение о том, что очередь не была создана ранее	Ожидание следующего действия.
17	Выбор пользователя равен 1 Пункт меню 8 Очередь создана ранее и не пустая	Сообщение об успешном удалении элемента из очереди	Ожидание следующего действия.
18	Выбор пользователя равен 1 Пункт меню 8 Очередь не создана ранее или пустая	Сообщение о том, что очередь либо не создана, либо пустая	Ожидание следующего действия.
19	Выбор пользователя равен 1 Пункт меню 9 Очередь создана ранее и не пустая	Вывод очереди на экран	Ожидание следующего действия.

20	Выбор пользователя равен 1 Пункт меню 9 Очередь не создана ранее или пустая	Сообщение о том, что очередь либо не создана, либо пустая	Ожидание следующего действия.
21	Выбор пользователя равен 1 Пункт меню 10 Очередь не создана ранее или пустая	Сообщение о том, что очередь еще не была создана	Ожидание следующего действия.
22	Выбор пользователя равен 1 Пункт меню 10 Очередь создана ранее	Сообщение об успешном очищении очереди	Ожидание следующего действия.
23	Выбор пользователя равен 1 Пункт меню 11 Массив адресов пустой	Сообщение о том, что массив ранее освобожденных адресов пустой	Ожидание следующего действия.
24	Выбор пользователя равен 1 Пункт меню 11 Массив адресов не пустой	Вывод массива адресов	Ожидание следующего действия.
25	Выбор пользователя равен 1 Пункт меню 12	Вывод сравнений времен работы различных функций для работы с очередью	Ожидание следующего действия.
26	Выбор пользователя равен 1 Пункт меню 13	Вывод сравнения преставления в памяти очереди в виде списка и массива	Ожидание следующего действия.
27	Выбор пользователя равен 0 Пункт меню 5 Моделирование очереди в виде списка не проводилось	Сообщение о том, что массив адресов пустой	Ожидание следующего действия.
28	Выбор пользователя равен 0 Пункт меню 5 Моделирование очереди в виде списка проводилось	Вывод массива адресов на экран	Ожидание следующего действия.

Негативные тесты

№ Теста	Входные данные	Выходные данные	Результат
01	Начало работы, выбор пользователя равен -10	Сообщение о том, что введен неверный номер выбора	Завершение программы с ненулевым кодом возврата
02	Начало работы, выбор пользователя равен 2	Сообщение о том, что введен неверный номер выбора	Завершение программы с ненулевым кодом возврата
03	Выбор пользователя равен 0 Пункт меню 1 Ввод не валидных значений времен обработки	Сообщение о том, что введен неверные значения времен обработки	Завершение программы с ненулевым кодом возврата
04	Выбор пользователя равен 0 Пункт меню 2 Ввод не валидных значений времен добавления	Сообщение о том, что введен неверные значения времен добавления	Завершение программы с ненулевым кодом возврата
05	Выбор пользователя равен 0 Пункт меню 3 или 4 Переполнение очереди при моделировании	Сообщение о том, что при моделировании СМО произошло переполнение очереди	Завершение программы с ненулевым кодом возврата
06	Выбор пользователя равен 1 Пункт меню 1 Размерность очереди равна -1	Сообщение о том, что введен не валидный размер очереди	Завершение программы с ненулевым кодом возврата
07	Выбор пользователя равен 1 Пункт меню 1 Размерность очереди равна 10001	Сообщение о том, что введен не валидный размер очереди	Завершение программы с ненулевым кодом возврата
08	Выбор пользователя равен 1 Пункт меню 1 Размерность очереди равна 1ваапа	Сообщение о том, что введен не валидный размер очереди	Завершение программы с ненулевым кодом возврата

9. Ответы на контрольные вопросы

9.1. Что такое FIFO и LIFO?

Структурой данных, которая работает по принципу FIFO, то есть первым зашёл - первым вышел, является очередь. Элемент добавляется в хвост очереди, а выходит с головы очереди.

Структурой данных, которая работает по принципу LIFO, то есть последним пришел -первым вышел, является стек.

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

- При хранении массивом:

  кол-во элементов * размер одного элемента очереди.

  Память выделяется на стеке при компиляции, если массив статический, либо 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.

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

При хранении очереди массивом память не освобождается, а просто меняется указатель на конец очереди и освобождается при завершении программы.

При хранении очереди списком, при удалении элемента очереди память, выделенная под его хранение, освобождается.

9.4. Что происходит с элементами очереди при ее просмотре?

При классической реализации элементы очереди удаляется при ее просмотре.

9.5. От чего зависит эффективность физической реализации очереди?

При реализации очереди в виде массива, может возникнуть переполнение, фрагментации нет. Быстрее работают операции добавления и удаления элементов.

При реализации очереди в виде списка легче удалять и добавлять элементы. Размер очереди ограничен лишь оперативной памятью компьютера. Может возникнуть фрагментация памяти.

Если размер очереди заранее не известен, то луче воспользоваться списком. Иначе – массивом.

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

При реализации очереди в виде массива не возникает фрагментация памяти, так же может возникнуть переполнение очереди, и тратиться дополнительное время на сдвиги элементов (классический массив). Сдвигов нет, если использовать кольцевой статический массив, но усложняется реализация алгоритмов добавления и удаления элементов.

При реализации очереди в виде списка проще выполнять операции добавления и удаления элементов, но может возникнуть фрагментация памяти.

9.7. Что такое фрагментация памяти, и в какой части ОП она возникает?

Фрагментация – это дробление памяти на мелкие не смежные свободные области маленького размера. Фрагментация возникает после выполнения системой большого числа запросов на память, таких, что размеры подходящих свободных участков памяти оказываются немного больше, чем требуемые.

Возникает в динамической части ОП.

9.8. Для чего нужен алгоритм «близнецов».

Метод близнецов (*или buddy system*) является схемой выделения памяти, сочетающей в себе возможность слияния буферов, и распределитель по степени числа 2. В основе метода лежит создание буферов малого размера путем деления пополам больших буферов и слияния смежных буферов по мере возможности. При разделении буфера на два каждая половина называется близнецом (buddy) второй.

Использование алгоритма «близнецов» для ускорения работы программы, то есть увеличение эффективности, при этом он уменьшает вероятность фрагментации памяти и улучшает поиск блок памяти для выделения.

9.9. Какие дисциплины выделения памяти вы знаете?

- Первый подходящий

Выбирается первая найденная подходящая память для выделения под данный элемент. При этом является более эффективным по поиску выделение памяти.

- Самый подходящий

Выбирается из всей возможной памяти самая «лучшая» подходящая память для выделения, но при этом вероятность нахождения такой памяти мала, поэтому бывают случай, когда выделяется память, так что образуются пустые места в памяти, то есть выделение памяти, больше, чем ожидалось.

9.10. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на корректное выделение и освобождение динамически выделенной памяти. Также необходимо проверить правильность реализации операций для кольцевой очереди.

9.11. 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 указного блока программы, ОС убирает его из таблицы адресов, однако указатель на этот блок может остаться в программе. Обращение к этому адресу и попытка считать данные из этого блока могут привести к неопределенному поведению, так как данные могут быть уже изменены.