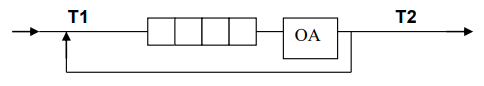
**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**«ОБРАБОТКА ОЧЕРЕДЕЙ»**

1. описание условия задачи (Вариант-7):

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 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.

2) техническое задание

* описание исходных данных и результатов (то есть, типы, форматы, точность, способ передачи, ограничения):

Исходные данные:

* выбор пункта меню (целое число от 0 до 4 включительно):

1 - обработать заявки с помощью статического массива

2 - обработать заявки с помощью односвязного списка

3 - сравнить реализации статическим массивом и односвязным списком

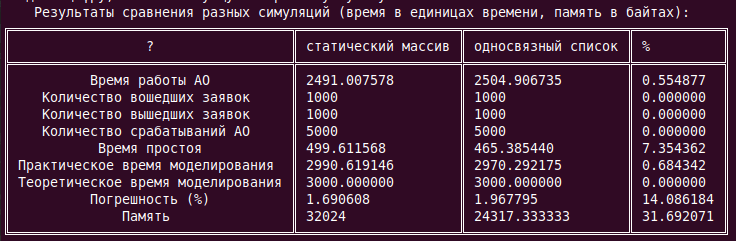
4 - изменить начальные значения системы

0 - завершить выполнение программы

* по требованию пользователя выдаются на экран адреса элементов очереди при их удалении и добавлении (1 – выводить, 0 – нет);
* настройки начальных значений системы (пункт меню 4) - целое число с нижним ограничением;

Выходные данные:

* промежуточные отчёты о текущей заявке через равные промежутки времени (время прихода заявки (в е.в.), время обслуживания заявки (в е.в.), количество которое данная заявка уже прошла обслуживающий аппарат, средняя длина очереди).
* полный отчёт по завершении работы симулятора (время работы аппарата (в е.в.), количество вошедших в систему заявок, количество вышедших из системы заявок, количество срабатываний ОА, время простоя аппарата (в е.в.), практическое время моделирования (в е.в.), теоретическое время моделирования (в е.в.), погрешность (в %)).
* таблица сравнения симуляторов, работающих на статическом массиве и односвязном списке (время в единицах времени, занимаемая память в байтах, процентное сравнение)



* описание задачи, реализуемой программой:

Система массового обслуживания, состоящую 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 вновь поступает в 'хвост' очереди, совершая всего 5 циклов обслуживания, после чего покидает систему. В начале процесса в системе заявок нет.

Программа реализует 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

- количество вошедших в систему и вышедших из нее заявок

- количество срабатываний ОА

- время простоя аппарата.

По требованию пользователя выдаются на экран адреса элементов очереди при их удалении и добавлении.

Меню:

1 - обработать заявки с помощью статического массива

2 - обработать заявки с помощью односвязного списка

3 - сравнить реализации статическим массивом и односвязным списком

4 - изменить начальные значения системы

0 - завершить выполнение программы

* способ обращения к программе:

Для запуска программы необходимо в терминале перейти в папку, содержащую исполняемый файл программы, и запустить его командой ./app.exe. Дальнейшее взаимодействие с программой происходит также в терминале.

* описание возможных аварийных ситуаций и ошибок пользователя:

|  |  |
| --- | --- |
| ***Ошибка пользователя*** | ***Ответ программы*** |
| Некорректный ввод целого числа в диапазоне: введено нецелое число / число не из диапазона / не число | Обнаружение ошибки, вывод сообщения о некорректном вводе и повторный запрос:  Например: *Ошибка ввода! Введите цифру от 0 до 4:* |
| Переполнение очереди при использовании статического массива. | Вывод сообщения об ошибке, прерывание симулятора и возвращение к меню.  *Статическая очередь переполнена! Работа симулятора прервана* |

1. описание внутренних структур данных

// таблица

ft\_table\_t \*table = ft\_create\_table();

//что выбрал пользователь в меню (целое число)

int user = NEITRAL\_USER;

//код возврата (целое число)

int rc = OK;

// коды ошибок

#define OK 0

#define ERROR 1

// эпсилон

#define EPS 1e-8

// макрос минимум

#define MIN(x,y) (((x) < (y)) ? (x) : (y))

// количество замеров

#define COUNT\_SORT 15

// нейтральное значение числа - выбранного пункта меню

#define NEITRAL\_USER -1

// ошибочное значение числа - выбранного пункта меню

#define ERROR\_USER -2

#define MAX\_COUNT\_WORK1 5

#define MAX\_COUNT\_APP 1000

#define REPORT\_COUNT\_APP 100

// Структура Заявка

struct Applic

{

double time\_come; // Время прихода

double time\_work; // Время обслуживания

int count\_work; // Количество проходов аппарата

};

// Структура ОА (обрабатывающий аппарат)

struct AO

{

double time\_stay; // Время простоя

int count\_work; // Количество заявок

};

// Структура Данные об очереди

struct Inf\_oueue

{

int len\_q\_now; // Текущая длина очереди

int len\_q\_max; // Максимальная длина очереди

int count\_change\_q; // Количество изменений очереди

int len\_q\_sum; // Сумма всех длин очередей

int count\_come\_app; // Количество вхождений заявок

int count\_away\_app; // Количество заявок, вышедших из системы

double time\_sim; // Общее время симуляции

int ind\_q\_file\_add; // Индекс элемента, который добавляется из хранилища в очередь

};

// Структура - элемент односвязного списка

struct Node

{

struct Applic \*data; // Указатель на структуру - заявку

struct Node \*next; // Указатель на следующий элемент односвязного списка

};

1. описанный алгоритм:

При запуске программы выводится информация о ней и меню, запрашивается номер выбранного пункта, пока не будет введено валидное число.

Если пользователь ввёл число 1, вызывается функция реализующая симулятор очереди и обработчика заявок, используя статический массив. Создаётся массив, в котором заранее рандомно генерируются времена прихода и времена обработки заявок, и статический массив - очередь. После этого запускается цикл, в котором, пока из системы не выйдет 1000 заявок, будет проверяться:

а) если пришла новая заявка, добавляем её в очередь;

б) если аппарат свободен очередь не пустая, достаём заявку из очереди и отправляем её на обработку аппарату.

После этого все времена в системе обновляются (время симуляции, работы и простоя аппарата, обработки и прихода текущих заявок), уменьшаясь на минимальное время из времени обработки и прихода текущих заявок, если аппарат занят, или на время прихода текущей заявки, если аппарат свободен. Также через каждые 100 заявок выводится частичный отчет о текущем состоянии (Текущая заявка: время прихода (в е.в.), время обслуживания (в е.в.), сколько раз уже была обслужена; средняя длина очереди), а после обработки всех заявок выводится полный отчёт о работе (время работы аппарата (в е.в.), количество вошедших в систему заявок, количество вышедших из системы заявок, количество срабатываний ОА, время простоя аппарата (в е.в.), практическое время моделирования (в е.в.), теоретическое время моделирования (в е.в.), погрешность (в %)). Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

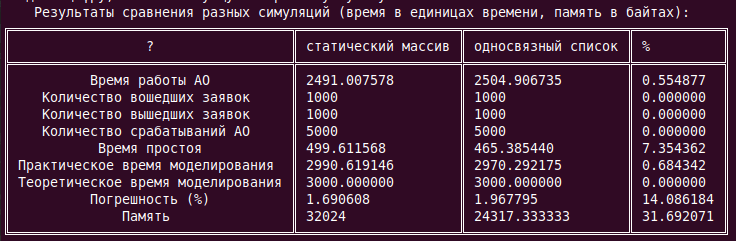
Если пользователь ввёл число 2, вызывается функция реализующая симулятор очереди и обработчика заявок, используя односвязный список. Создаётся массив, в котором заранее рандомно генерируются времена прихода и времена обработки заявок, и указатели на голову односвязного списка-очереди и на хвост. После этого запускается цикл, в котором, пока из системы не выйдет 1000 заявок, будет проверяться:

а) если пришла новая заявка, добавляем её в очередь;

б) если аппарат свободен очередь не пустая, достаём заявку из очереди и отправляем её на обработку аппарату.

После этого все времена в системе обновляются (время симуляции, работы и простоя аппарата, обработки и прихода текущих заявок), уменьшаясь на минимальное время из времени обработки и прихода текущих заявок, если аппарат занят, или на время прихода текущей заявки, если аппарат свободен. Также через каждые 100 заявок выводится частичный отчет о текущем состоянии (Текущая заявка: время прихода (в е.в.), время обслуживания (в е.в.), сколько раз уже была обслужена; средняя длина очереди), а после обработки всех заявок выводится полный отчёт о работе (время работы аппарата (в е.в.), количество вошедших в систему заявок, количество вышедших из системы заявок, количество срабатываний ОА, время простоя аппарата (в е.в.), практическое время моделирования (в е.в.), теоретическое время моделирования (в е.в.), погрешность (в %)). Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 3, вызывается функция вывода таблицы сравнения симуляторов с разными реализациями очереди. Программа несколько раз прогоняет симуляторы и делает замеры (памяти, времени, количества, погрешность), а затем находит среднее арифметическое найденных значений и выводит на экран табличку (пункт для сравнения | для статического массива | для односвязного списка | процентное сравнение). Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.



Если пользователь ввёл число 4, программа запрашивает целые числа в диапазоне для изменения начальных настроек системы: минимального времени прихода заявки (>=0), максимальное время прихода заявки (>=минимальному), минимального времени обработки заявки (>=0), максимальное время обработки заявки (>=минимальному). Затем вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 0, программа завершается.

5) набор тестов, с указанием, что проверяется:

|  |  |  |
| --- | --- | --- |
| Тестовые данные | Что проверяется | Вывод программы |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |  | файл |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |  | структура |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

6) оценка эффективности:

\* Время выводится в тактах процессора, память в байтах, количество замеров времени = 50 раз.

* 40 элементов (минимальное количество) уже отсортированы:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 214,680000 | 162,300000 | 21280,000000 |
| Таблица ключей | 17,320000 | 6,820000 | 21600,000000 |
| Выигрыш от таблицы ключей в процентах | 1139,49% | 2279,77% | -1,48% |

* 40 элементов (минимальное количество) расположены рандомно:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 235,400000 | 352,040000 | 21280,000000 |
| Таблица ключей | 18,480000 | 10,700000 | 21600,000000 |
| Выигрыш от таблицы ключей в процентах | 1173,80% | 3190,09% | -1,48% |

* 40 элементов (минимальное количество) уже отсортированы в обратном порядке:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 218,820000 | 251,920000 | 21280,000000 |
| Таблица ключей | 17,160000 | 9,720000 | 21600,000000 |
| Выигрыш от таблицы ключей в процентах | 1175,17% | 2491,77% | -1,48% |

* 70 элементов (среднее количество) уже отсортированы:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 437,580000 | 403,680000 | 37240,000000 |
| Таблица ключей | 38,020000 | 13,240000 | 37800,000000 |
| Выигрыш от таблицы ключей в процентах | 1050,92% | 2948,94% | -1,48% |

* 70 элементов (среднее количество) расположены рандомно:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 484,460000 | 788,580000 | 37240,000000 |
| Таблица ключей | 45,620000 | 20,400000 | 37800,000000 |
| Выигрыш от таблицы ключей в процентах | 961,95% | 3765,59% | -1,48% |

* 70 элементов (среднее количество) уже отсортированы в обратном порядке:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 395,800000 | 537,920000 | 37240,000000 |
| Таблица ключей | 46,040000 | 16,320000 | 37800,000000 |
| Выигрыш от таблицы ключей в процентах | 759,69% | 3196,08% | -1,48% |

* 100 элементов (максимальное количество) уже отсортированы:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 592,020000 | 622,300000 | 53200,000000 |
| Таблица ключей | 90,120000 | 20,200000 | 54000,000000 |
| Выигрыш от таблицы ключей в процентах | 556,92% | 2980,69% | -1,48% |

* 100 элементов (максимальное количество) расположены рандомно:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 912,940000 | 1510,040000 | 53200,000000 |
| Таблица ключей | 127,840000 | 200,640000 | 54000,000000 |
| Выигрыш от таблицы ключей в процентах | 614,13% | 652,61% | -1,48% |

* 100 элементов (максимальное количество) уже отсортированы в обратном порядке:

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Сорт. выбором | Рекурсивная сорт. | Объем памяти |
| Полная таблица | 633,820000 | 929,500000 | 53200,000000 |
| Таблица ключей | 81,380000 | 23,360000 | 54000,000000 |
| Выигрыш от таблицы ключей в процентах | 678,84% | 3879,02% | -1,48% |

7) выводы по проделанной работе:

Лабораторная работа №2 «Записи с вариантами. Обработка таблиц» помогла осознать, что для хранения записей с большим количеством полей можно использовать специальные типы данных (например, структуры в Си), хранящие информацию об одном объекте как единое целое. А для хранения информации разных типов в одном месте памяти также удобно использовать специальные типы данных (например, объединения в Си). Эти типы можно комбинировать и вкладывать друг в друга в зависимости от задачи.

В данной задаче программа, использующая таблицу ключей для сортировки, занимает на 1,48% памяти больше, но при этом работает во много раз быстрее.

8) ответы на вопросы:

1. Что такое FIFO и LIFO?

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

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

4. Что происходит с элементами очереди при ее просмотре?

5. От чего зависит эффективность физической реализации очереди?

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

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

8. Для чего нужен алгоритм «близнецов».

9. Какие дисциплины выделения памяти вы знаете?

10. На что необходимо обратить внимание при тестировании программы?

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

Отчёт подготовила Талышева Олеся из группы ИУ7-35Б