Отчёт по лабораторной работе №5

“Работа с очередью”

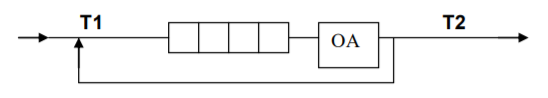
Вариант 7

Выполнил: Соколов Ефим Маркович (группа ИУ7-33Б)

### ***Цель работы:*** отработать навыки работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Провести сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценить эффективности программы (при различной реализации) по времени и по используемому объему памяти.

Техническое задание

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. 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.

Типы входных и выходных данных

***Входные данные:***

* Ключ меню – целое число в промежутке [0 ; 4] представляющее собой команду меню
* Интервалы характеристик обслуживающего аппарата (ОА)

**Выходные данные:**

* Результат моделирования работы ОА: число вошедших заявок, число вышедших заявок, число необработанных заявок, число срабатываний автомата, реальное рабочее время автомата, ожидаемое рабочее время автомата, временная погрешность, время простоя автомата, количество вновь использованных адресов и адресов, взятых из новой памяти (для реализации на списке))
* Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании связанных списков и массива

***Возможные аварийные ситуации:***

* Некорректный ввод характеристик ОА (введены нецелые числа, символы, отрицательные числа, нижний предел времени входа (обработки) заявки больше верхнего предела входа (обработки)
* Попытка моделирования очереди при уже имеющейся в программе

Структуры данных

// Хранение заявки

typedef struct

{

*int* num; // сколько раз заявка была обработана

*double* time\_out; // время, затрачиваемое на обработку заявки

} task\_t;

// Реализация очереди на основе массива

typedef struct

{

*int* capacity; // максимальный допустимый размер очереди

int size; // текущий размер очереди

int rear; // “хвост” очереди

int front; // “голова” очереди

*task\_t* \*array; // указатель на массив заявок

} qArray\_t;

// Реализация очереди на основе односвязного линейного списка

// Элемент очереди

typedef struct node

{

*task\_t* task; // заявка

*struct node* \*next; // указатель на следующий элемент (узел) очереди

} node\_t;

typedef struct

{

*int* capacity; // максимальный допустимый размер очереди

int size; // текущий размер очереди

*node\_t* \*rear; // указатель на “хвост” очереди

node\_t \*front; // указатель на “голову” очереди

} qList\_t;

// Реализация массива свободных областей памяти

typedef struct

{

*size\_t* \*array; // указатель на массив адресов памяти

*int* capacity; // максимальный размер массива

*int* idx; // текущий незанятый элемент

} array*\_t*;

Алгоритм

1. При выборе команды вывода статистики ОА, выводится промежуточная статистика ОА (число обработанных заявок, среднее и текущий размеры очереди) после обработки каждой 100-ой заявки, а также общие данные, описанные в секции *“Выходные данные”,* после обработки первой 1000 заявок.
2. При выборе команды вывода сравнительного анализа выполнения операций над очередью, выводится среднее значение времени добавления (удаления) элементов из очереди на основе 1000 добавлений (удалений).

Тесты

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Тест | | Пользовательский ввод | | Результат | |
| 1 | Некорректный ввод команды меню | | -7 | | Сообщение “Введен недопустимый ключ! Повторите попытку” | |
| 2 | Некорректный ввод характеристик ОА  (нецелочисленный ввод) | | Ввод интервала прихода:  0 aaa | | Сообщение “Введено недопустимое значение! Повторите попытку” | |
| 3 | Некорректный ввод характеристик ОА  (одна из границ отрицательна) | | Ввод интервала прихода:  -3 0 | | Сообщение “Введено недопустимое значение! Повторите попытку” | |
| 4 | Некорректный ввод характеристик ОА  (правая граница меньше левой) | | Ввод интервала выхода:  4 2 | | Сообщение “Введено недопустимое значение! Повторите попытку” | |
| 5 | Некорректный ввод характеристик ОА  (неположительное число повторений одной заявки) | | Ввод количества обслуживаний одной заявки:  -5 | | Сообщение “Введено недопустимое значение! Повторите попытку” | |
| 6 | Попытка создания очереди при уже существующей | | Попытка  моделирования  очереди | | Сообщение “Очередь уже была смоделирована” | |
| 7 | Тривиальный  тест | | 0 6  0 1  5 | | Вывод информации ОА и количественная характеристика моделирования | |
| 8 | Переполнение очереди | | Переполнение очереди во время работы ОА | | Инкрементация счетчика необработанных заявок | |
| 9 | Вывод количественной характеристики выполнения операций над очередью | | Выбор соответствующей команды меню | | Добавление элементов в очередь на основе: T1  Удаление элементов из очереди на основе: T2 | |

Оценка эффективности

**Время\* добавления элемента (в тактах процессора):**

|  |  |
| --- | --- |
| Массив | Список |
| 51 | 348 |

**Время\* удаления элемента (в тактах процессора):**

|  |  |
| --- | --- |
| Массив | Список |
| 39 | 164 |

**Занимаемая память (в байтах)\*\*:**

|  |  |  |
| --- | --- | --- |
| Количество элементов | Массив | Список |
| 10 | 144 | 224 |
| 100 | 1224 | 2024 |
| 1000 | 12024 | 20024 |

\*Среднее количество тактов, полученное по результатам 1000 замеров времени

\*\*Указанные результаты измерения занимаемой памяти справедливы для заранее выделенного объёма памяти

Вывод

Использование связанных списков невыгодно при реализации очереди: они проигрывают как по памяти, так и по времени обработки. Однако, когда заранее неизвестен максимальный размер очереди, то выгоднее использовать связанные списки, поскольку в отличие от статического массива, списки ограничены в размерах только размером оперативной памяти машины. Также использование списков позволяет добиться фрагментации памяти, что полезно при относительной “забитости” памяти – невозможности выделить непрерывный достаточно большой участок памяти.

Контрольные вопросы

* 1. **Что такое очередь?**

Очередь - структура данных, для которой выполняется правило FIFO (First In First Out - первым зашёл - первым вышел. Вход находится с одной стороны очереди, выход - с другой.

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

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

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

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

При хранении списком, память под удаляемый кусок освобождается.

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

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

Эти элементы удаляются из очереди.

* 1. **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. Также реализация связанным списком позволяет добиться фрагментации памяти, что полезно в определенных случаях.

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

При реализации статическим кольцевым массивом, очередь всегда ограничена по размеру, но операции выполняются быстрее, нежели чем на списке.

При использовании линейного списка тратится больше времени на обработку операций с очередью, а также может возникнуть фрагментация памяти.

* 1. **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 участков памяти.

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

При тестировании программы необходимо обратить внимание на возникновение фрагментации памяти, 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.

* 1. **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.