|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 5 |

**Название:**

Обработка очередей

**Дисциплина:** Типы и структуры данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-36Б |  | | А.А. Жаворонкова |
|  | (Группа) | |  | (И.О. Фамилия) |
|  |  | |  |  |
| Преподаватель: | Никульшина Т. А. | |  |  |

Москва, 2022

Описание условия задачи

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.

Описание ТЗ

*Описание исходных данных и результатов*

*Исходные данные:* пункт меню (см. описание задачи), времена прихода и обработки заявок двух типов.

*Результаты:* ожидаемое время моделирования, реальное время моделирования (в единицах времени и наносекундах), погрешность моделирования, время простоя аппарата, количество вошедших (вышедших) в систему заявок 1 (2) типа, адреса освобождаемых элементов.

*Описание задачи, реализуемой программой*

Меню:

1. Моделирование очереди в виде массива
2. Моделирование очереди в виде списка
3. Вывести адреса освобождаемых элементов
4. Изменить времена прихода и обработки
5. Вывести меню
6. Выйти из программы

*Способ обращения к программе*

Вызов программы происходит через терминал (main.exe). Дальнейшая работа с программой выполняется при помощи меню, выводимого на экран.

*Описание возможных ошибок пользователя*

* Ввод неверного пункта меню
* Вывод адресов освобождаемых элементов без моделирования в виде списка
* Некорректный интервал времени прихода и обработки

Описание внутренних структур данных

Очередь в виде кольцевого массива реализована следующей структурой:

typedef struct

{

    int arr[N];

    int pin;

    int pout;

    int len;

} queue\_arr;

**arr** – кольцевой массив, представляющий очередь  
**pin** – индекс начала очереди  
**pout** – индекс конца очереди  
**len** – длина очереди

Очередь в виде списка реализована следующими структурами:

typedef struct

{

    node\_t \*pin;

    node\_t \*pout;

    int len;

} queue\_list;

**pin** – указатель на начало очереди  
**pout** – указатель на конец очереди  
**len –** длина очереди

Каждый элемент очереди представлен такой структурой:

typedef struct node

{

    int x;

    struct node \*next;

} node\_t;

**x** – значение элемента очереди  
**next** – указатель на следующий элемент

Набор тестов с указанием проверяемого параметра

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Что проверяется |
| -1 1 | «Введены некорректные значения» | Ввод отрицательной границы интервала |
| 5 5 | «Введены некорректные значения» | Ввод совпадающих границ интервала |
| 0 4 | Новый интервал | Ввод корректного интервала |
| 10 | «Введены некорректные данные» | Ввод неверного пункта меню |
| 3 | «Ни одна область еще не была освобождена» | Вывод адресов освобождаемых элементов без моделирования очереди в виде списка |
| 1 | Ожидаемое время моделирования, реальное время моделирования (в единицах времени и наносекундах), погрешность моделирования, время простоя аппарата, количество вошедших (вышедших) в систему заявок 1 (2) типа, адреса освобождаемых элементов | Моделирование очереди в виде массива |

Описание алгоритма

Как только приходит заявка (первого или второго типа), она помещается в соответствующую очередь. В тот же момент рассчитывается через сколько единиц времени придет заявка этого же типа.   
Как только обслуживающий аппарат отработал, проверяется наличие заявок в первой очереди (наиболее приоритетной). Если заявки в ней есть, из очереди «достается» первая и рассчитывается время, через которое автомат отработает. Если в первой очереди нет заявок, а во второй есть, то аналогичные действия проводятся для заявки второго типа.  
Если же автомат отработал, но обе очереди пустые, обслуживающий аппарат ожидает первую пришедшую заявку, а время ожидания считается временем простоя ОЭ.

Временная эффективность и затраты памяти

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Длина очереди | Массив | | | Список | | |
| Память, б | Время, мс | Ошибка | Память, б | Время, мс | Ошибка |
| 150 | 8024 | 70.649500 | 1.840 % | 2448 | 114.446600 | 0.230 % |
| 242 | 8024 | 196.508000 | 0.094 % | 3920 | 288.316200 | 1.137 % |
| 419 | 8024 | 372.371200 | 0.492 % | 6752 | 515.272200 | 0.466 % |

Как можно заметить из таблицы, массив имеет преимущество перед списком по времени. Выигрыш по времени получается потому, что в случае списка при добавлении элемента в очередь необходимо выделять память под него. Выигрыш массива по памяти возможен при достаточно большой очереди. В данном случае был выделен массив на 10000 элементов (8024 б), а длина очереди не превышала 500 элементов: большая часть выделенной памяти оказалась неиспользованной.   
Если выделить массив на 500 элементов, то при очереди более, чем 150 элементов массив будет иметь преимущество, то есть при заполненности массива более, чем на 30%.

Вывод

Сравнивая полученные результаты с результатами ЛР №4, можно сделать вывод, что структуры данных, работающие по принципу LIFO и FIFO, имеют разные преимущества в зависимости от их реализации.

При реализации стека и очереди в виде массива у них появляется преимущество по времени. Преимущество по памяти возможно при условии определенного процента заполненности (30% для типа “int”).

При реализации стека и очереди в виде списка их размеры ограничиваются только размерами оперативной памяти, что позволяет создавать их большего размера.

Ответы на вопросы

1. Что такое FIFO и LIFO?

FIFO - способ организации доступа к данным по принципу «первый вошёл - первый вышел», LIFO – «последний вошёл - первый вышел».

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

В случае реализации очереди на основе массива, объём выделяемой памяти под изначальный массив определяется заранее. В случае реализации очереди на основе списка память выделяется отдельно под каждый элемент.

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

В случае реализации очереди на основе массива, память при удалении элементов не освобождается, а только сдвигается указатель на начало очереди, в случае реализации очереди на основе списка, при удалении элемента, освобождается память, выделенная под узел, его содержащий.

1. Что происходит с элементами очереди при ее просмотре?

При просмотре элементов очереди они удаляются.

1. От чего зависит эффективность физической реализации очереди?

Эффективность физической реализации очереди зависит от эффективности её расположения в памяти: операции с адресной арифметикой довольно быстры, в то время как необходимость проходить весь список - медленная операция.

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

Достоинство массива – скорость.

Достоинство списка – неограниченность по памяти.

Недостатки массива – ограниченность по память, фиксированная память.

Недостатки списка – возможность фрагментации, скорость выполнения операций.

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

Фрагментация памяти - явление, при котором информация записывается в

использованных и освобожденных блоках памяти, таким образом она

находится в разных местах доступной памяти. В результате появляются

большое количество свободных участков памяти небольшого размера,

расположенных разрозненно.

1. Для чего нужен алгоритм «близнецов».

Алгоритм «близнецов» нужен для наиболее эффективного использования памяти: возможность разбивать участки памяти на более мелкие или объединять их. Преимуществом этого алгоритма является скорость, но его реализация усложняется за счет необходимости вести систему списков свободных блоков.

1. Какие дисциплины выделения памяти вы знаете?

Две основные дисциплины памяти - выделение «самого подходящего» блока – наиболее приближенного к запрашиваемому по размеру – и «первого подходящего» – первого блока, достаточного для выделения требуемого размера.

1. На что необходимо обратить внимание при тестировании программы?

На возможное переполнение очередей, на следующие ситуации: время прихода и обработки первой заявки меньше соответствующих времен для второй заявки (в таком случае заявки второго типа практически не будут обслуживаться), аналогичная обратная ситуация (заявок второго типа будет обработано примерно вдвое больше, чем первого типа).

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

При динамическом распределении памяти объекты размещаются в куче: при создании объекта указывается размер запрашиваемой под объект памяти, и, в случае успеха, выделенная область памяти становится недоступной при последующих операциях выделения памяти. Противоположная по смыслу операция — освобождение занятой ранее под какой-либо объект памяти: освобождаемая память возвращается в кучу и становится доступной при дальнейших операциях выделения памяти.