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

О Т Ч Е Т

По лабораторной работе № 5

Название: Обработка очередей

Дисциплина: Типы и структуры данных

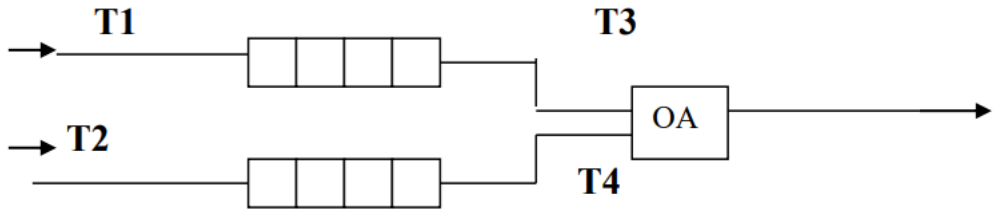
Студент ИУ7-32Б Беляк С.С.

Преподаватель Барышникова М.Ю.

Москва, 2023

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 T1 и T2, 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 T3 и T4, 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 немедленно поступает на обслуживание; обработка заявки 2- го типа прерывается, и она возвращается в "хвост" своей очереди (система с абсолютным приоритетом и повторным обслуживанием). Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок 1-го типа информацию о текущей и средней длине каждой очереди. В конце процесса выдать общее время моделирования и количестве вошедших в систему и вышедших из нее заявок обоих типов, среднем времени пребывания заявок в очереди, количестве «выброшенных» заявок второго типа. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и ОА добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Описание задачи, реализуемой в программе

Цель работы - приобрести 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 указанных структур данных, оценить эффективности программы по времени и по используемому объему памяти.

Описание ТЗ.

Описание исходных данных

Пользователь выбирает опцию из меню, представленного в консоли.

Возможные варианты выбора:

Меню:

1. Запустить симуляцию для очереди массивом.
2. Запустить симуляцию для очереди списком.
3. Оценка эффективности.
4. Ручной режим.
5. Изменить параметры симуляции.
0. Выйти из меню.

Пользователь может выбрать соответствующую опцию, введя число от 0 до 5 в консоли.

Имеются следующие ограничения:

Ограничение по памяти:

Программа может использовать память для хранения очередей. Необходимо обеспечить эффективное использование памяти и предотвратить утечки.

Ограничение по времени:

Время выполнения программы может зависеть от множества факторов, включая параметры симуляции и объем обрабатываемых данных.

Ограничение по количеству заявок:

В симуляции предусмотрено количество вышедших заявок, равное 1000. Это ограничение может быть изменено.

Ограничение по количеству операций ввода-вывода:

Программа взаимодействует с пользователем через стандартный ввод/вывод. Необходимо предусмотреть обработку ошибок при вводе и выводе данных.

Ограничения по точности моделирования:

Параметры симуляции могут влиять на точность моделирования. Необходимо внимательно выбирать значения параметров для достижения реалистичных результатов.

Ограничения на структуры данных:

Программа использует структуры данных для представления очередей. Важно обеспечивать их правильное использование и обработку.

Ограничения по вводу параметров:

Пользователь должен вводить параметры симуляции. Программа должна корректно обрабатывать ввод пользователя и предоставлять информацию о возможных ошибках.

1. Описание результатов программы

1. **Запустить симуляцию для очереди массивом:**
    - Программа моделирует процесс обработки заявок в очереди, представленной массивом.
    - Выводит статистику по вышедшим, вошедшим, неудачным заявкам, текущей длине очереди, средней длине очереди и среднему времени ожидания в очереди.
    - Продолжает моделирование до достижения 1000 вышедших заявок.
2. **Запустить симуляцию для очереди списком:**
    - Моделирует процесс обработки заявок в очереди, представленной списком.
    - Выводит статистику по вышедшим, вошедшим, неудачным заявкам, текущей длине очереди, средней длине очереди и среднему времени ожидания в очереди.
    - Продолжает моделирование до достижения 1000 вышедших заявок.
3. **Оценка эффективности:**
    - Оценивает эффективность операций добавления и удаления заявок для очередей массивом и списком.
    - Выводит среднее время выполнения операций и объем занимаемой памяти для каждой структуры данных.
4. **Ручной режим:**
    - Позволяет пользователю вручную добавлять и удалять заявки в очереди массивом и списком.
    - Выводит адрес добавленного или удаленного элемента в соответствующей очереди.
    - Возможность просмотра текущего состояния очередей.
5. **Изменить параметры симуляции:**
    - Запрашивает у пользователя ввод параметров для симуляции, таких как интервалы времени доставки и выполнения заявок для двух очередей.
6. **Выйти из меню:**
    - Завершает выполнение программы.

2. Способ обращения к программе

Способ обращения к программе пользователем происходит через исполняемый файл app.exe.

3. Описание возможных аварийных ситуаций и ошибок пользователя

CHOICE_ERROR (Ошибка выбора)

Эта ошибка возникает, когда пользователь вводит некорректное значение в меню программы. Например, если пользователь вводит символ или значение, которое не соответствует доступным опциям в меню. Пользователю будет предложено ввести корректное значение из доступных опций.

ALLOCATE (Ошибка выделения памяти)

Происходит, когда очередь достигает своей максимальной длины. В случае ошибки переполнения, программа сообщит об этом пользователю, и вновь созданная заявка не будет добавлена в очередь.

OVERFLOW (Ошибка переполнения)

Возникает, когда система не может выделить достаточно памяти для новой заявки. В случае ошибки выделения памяти, программа сообщит об этом пользователю, и заявка не будет добавлена в очередь.

5.Описание внутренних структур данных

Структуры в программе служат для представления и управления очередей на основе массива и очередей на основе списка:

Структура **SimulationLog:** Структура для записи симуляции. Записывает информацию о входящих, выходящих задачах, неудачных попытках и функциональных вызовах.

```
typedef struct
{
    int task_in_count;
    int task_out_count;
    int failed_task_count;
    int function_call_count;
    int total_length;
} SimulationLog;
```

Структура **arr_queue_t:** Структура для представления очереди с использованием массива. Хранит массив указателей на задачи. Имеет входной и выходной указатели, текущее количество, максимальную вместимость. Отслеживает выделенную память.

```
typedef struct{
    task_t **data;
    int Pin;
    int Pout;
    int amount;
    int maxamount;
    int allocated;
} arr_queue_t;
```

Структуры **node_t и list_queue_t:** Структура для представления очереди с использованием связанного списка. Узел хранит задачу и указатель на следующий узел. Отслеживает входной, выходной указатели, текущее количество и максимальную вместимость.

```
typedef struct node node_t;
struct node{
    task_t *task;
    node_t *next;
};
typedef struct{
    node_t *Pin;
    node_t *Pout;
    int amount;
    int maxamount;
} list_queue_t;
```

6. Алгоритм программы

Общий алгоритм программы:

1. Инициализация параметров симуляции (интервалы времени, связанные с задачами и очередями и структур данных для очередей (массив и список).
2. Основной цикл программы:
3. Программа входит в главный цикл, который ожидает ввода пользователя и предоставляет меню с различными опциями.
4. В зависимости от выбора пользователя, программа выполняет следующие действия:

- Запустить симуляцию для очереди массивом.
- Запустить симуляцию для очереди списком.
- Оценка эффективности.
- Ручной режим.
- Изменить параметры симуляции.
- Выход из программы.

5. Возврат в главное меню:
6. При завершении программы, программа освобождает выделенную память для структур данных, завершает работу и закрывает все ресурсы.

Вывод по алгоритму работающей программы:

Алгоритм программы подразумевает эффективное управление памятью, выделение и освобождение ресурсов в соответствии с операциями над очередями.

Достоинства различных реализаций очередей обусловлены требованиями к конкретным операциям и особенностями использования памяти.

В целом, программа представляет собой систему моделирования с удобным пользовательским интерфейсом, эффективным управлением памятью и возможностью детального анализа характеристик симулируемой системы.

Временная эффективность и затраты памяти

Найдем среднее время и затраты памяти для функций создания и удаления очередей при 10000 итерациях:

Создание и удаление очередей:

Тип данных	Время (мс)	Память(б)
Массив	1006	4020
Список	1006	16008

Найдем среднее время и затраты памяти для функций симуляции при 10000 итерациях:

Запуск симуляций:

Тип данных	Время (мс)	Память(б)
Массив	839	4020
Список	1006	16008

Среднее время и затраты памяти для функций создания и удаления очередей демонстрируют, что массивная реализация требует меньше памяти по сравнению со связанным списком. Однако, время создания и удаления оказывается одинаковым для обоих структур.

В контексте запуска симуляций, время выполнения для массивной реализации симуляции снижается по сравнению со временем создания и удаления.

В свою очередь, для связанного списка время выполнения симуляции остается стабильным. При этом, затраты памяти остаются в том же порядке, где связанный список требует больше ресурсов, чем массивная реализация.

Теоретические расчёты времени моделирования.

1. Время обработки заявки:
    - Заявка 1-го типа обрабатывается 4 раза с временем от 0 до 4 е.в.
    - Среднее время обработки одной заявки 1-го типа: 2 е.в.
2. Время обработки заявки 2-го типа:
    - Заявка 2-го типа обрабатывается 1 раз с временем от 0 до 1 е.в.
    - Среднее время обработки одной заявки 2-го типа: 0.5 е.в.
3. Время прихода заявок:
    - Заявки 1-го типа поступают с интервалом от 1 до 5 е.в., среднее значение (1+5)/2=3(1+5)/2=3 е.в.
    - Заявки 2-го типа поступают с интервалом от 0 до 3 е.в., среднее значение (0+3)/2=1.5(0+3)/2=1.5 е.в.
4. Общее время моделирования:
    - Так как ОА работает с простоем, то общее время моделирования будет определяться временем прихода заявок. В данном случае, это 3×1000=3=3000 е.в.
5. Количество "выброшенных" заявок второго типа:
    - Заявка 2-го типа может быть вытеснена, если в момент обслуживания пустой очереди входит заявка 1-го типа. Количество таких случаев будет зависеть от конкретных условий моделирования.
6. Сравниваем практические результаты с теоретическими расчетами.

```
Время ожидания: 25.554867
Время работы: 2991.912598
Общее время моделирования : 3017.467529
Ожидаемое теоретическое время: 3000.000000
Погрешность: 0.58%
Заявки 1-го типа: Вошедших: 1001, Вышедших: 1000, Неудачных: 0
Заявки 2-го типа: Вошедших: 2002, Вышедших: 1986, Неудачных: 0
Среднее время пребывания в очереди 1: 3.017467
Среднее время пребывания в очереди 2: 1.519369
```

Для проверки правильности работы системы по входу общее время моделирования делим на время прихода одной заявки:

3017.467 / 3 = 1005.822 заявок

Таким образом, определяем предполагаемое количество вошедших заявок.

Фактически их пришло 1001, т.е. погрешность составляет

100% (1001-1005.828)/ 1005.828= 0.578%

Проверим правильность работы системы по выходу.

За время моделирования аппарат работал 2991.912 е.в. и простаивал – 25.554 е.в.

Время моделирования должно было составить: 3017,467 е.в., а оно составило – 3017,466 е.в.

Получаем погрешность:

100% (3017,467 –3017,466)/ 3017,466 = 0.00003%.

Теоретические расчеты времени моделирования и практические результаты позволяют сделать вывод о том, что созданная система симуляции в целом соответствует теоретическим ожиданиям.

Анализ фрагментации.

Важным аспектом является отсутствие фрагментации памяти при добавлении и удалении элементов. Адреса, выделенные для новых элементов, такие же, как и адреса, которые могли быть удалены. Это свидетельствует о том, что программа эффективно управляет памятью, избегая лишней фрагментации.

Вывод

В результате выполнения лабораторной работы была разработана программа, предназначенная для симуляции системы обработки заявок. Программа предоставляет удобное меню с различными опциями, включая возможность проведения симуляции, оценки эффективности, ручного режима и изменения параметров симуляции. В ходе симуляции программа выводит ключевые характеристики, такие как количество вошедших и вышедших заявок, неудачные заявки, текущая длина очереди и среднее время ожидания в очереди для обеих реализаций очередей: массивной и списочной.

Важным элементом лабораторной работы является оценка эффективности, включающая измерение времени выполнения операций и затраты памяти для каждого типа очереди. Программа предоставляет детальные описания структур данных, алгоритмов и обработки ошибок, что обеспечивает ясное понимание ее функционала.

Результаты работы подчеркивают значимость эффективного управления памятью и предоставляют удобный инструмент для анализа работы системы обработки заявок в различных сценариях.

Ответы на контрольные вопросы

1.Что такое FIFO и LIFO?

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).

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

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

Реализация очереди в виде массива. При моделировании простейшей линейной 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. В каждый текущий момент времени выделенная память может быть вся свободна, занята частично или занята полностью.

Реализация очереди в виде линейного списка. В этом случае в статической памяти можно либо хранить адрес начала и конца очереди, либо – адрес начала очереди и количество элементов.

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

В реализации с массивом: при удалении элемента из массивной очереди, выделенная под него память освобождается с использованием операции free. Важно обеспечить правильное управление индексами (Pin) и (Pout), чтобы избежать утечек памяти и корректно освободить выделенные ресурсы.

В реализации с использованием списка: при удалении элемента из списка, освобождается память, выделенная под сам элемент задачи, а также под узел списка. Освобождение памяти производится с использованием операции free.

4.Что происходит с элементами очереди при ее просмотре?

При просмотре элементов очереди массивом: происходит доступ к элементам массива по индексу, начиная с индекса Pout и заканчивая индексом Pin. Элементы при этом не удаляются из массива.

При просмотре элементов очереди списком: происходит перемещение по узлам списка с использованием указателей. Элементы не удаляются из списка в процессе просмотра.

5.От чего зависит эффективность физической реализации очереди?

Эффективность физической реализации очереди зависит от:

Частоты операций вставки и удаления: если операции вставки и удаления выполняются с примерно равной частотой, то реализация с использованием списка может быть эффективнее, так как добавление и удаление элементов в середине списка занимает постоянное время.

Объема данных: для больших объемов данных реализация с использованием массива может быть эффективнее из-за лучшей локальности данных и меньшего числа указателей.

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

Очередь массивом:

Достоинства: Простота реализации, хорошая локальность данных. Недостатки: Фиксированный размер, что может привести к переполнению или избыточной памяти.

Очередь списком:

Достоинства: Динамический размер, удобство вставки и удаления в середине очереди. Недостатки: больше затраты памяти на хранение указателей, возможны фрагментация памяти.

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

В процессе моделирования очереди может оказаться, что при последовательных запросах на выделение и освобождении памяти под очередной элемент выделяется не та память, которая была только что освобождена при удалении элемента. Участки свободной и занятой памяти могут чередоваться, т.е. может возникнуть фрагментация памяти.

8. Для чего нужен алгоритм «близнецов».

Алгоритм "близнецов" используется в симуляции с двумя очередями. Он предполагает, что при появлении нового элемента, необходимо решить, в какую из двух очередей его поместить. Если одна из очередей короткая, новый элемент помещается в эту очередь, иначе выбирается случайная очередь. Такой подход позволяет достичь баланса между двумя очередями и равномерно распределить элементы между ними.

9. Какие дисциплины выделения памяти вы знаете?

First Fit (Первый Подходящий): выделяет первый блок памяти, который подходит по размеру.

Best Fit (Лучший Подходящий): выделяет блок памяти, который наилучшим образом соответствует размеру запроса, минимизируя фрагментацию.

Worst Fit (Худший Подходящий): выделяет самый большой из доступных блоков памяти, что может привести к уменьшению количества оставшейся свободной памяти.

Next Fit (Следующий Подходящий): аналогичен First Fit, но начинает поиск с того места, где закончил предыдущий запрос.

Quick Fit (Быстрый Подходящий): использует несколько списков фиксированных размеров для быстрого доступа к блокам нужного размера.

Алгоритм "близнецов" (Twin Allocation).

10. На что необходимо обратить внимание при тестировании программы?

- Рассогласование между средними ожидаемыми временами и временами, полученными в моделирующей программе должно быть не больше 2–3%.
- Необходимо проверить правильность работы программы при различном заполнении очередей, т.е., когда время моделирования определяется временем обработки заявок и когда определяется временем прихода заявок;
- Отследить переполнение очереди, если очередь в программе ограничена.
- При реализации очереди списком необходимо тщательно следить за освобождением памяти при удалении элемента из очереди.
- Следует проверять, происходит ли при этом фрагментация памяти.

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

Выделение памяти:

При динамическом запросе памяти используется функция malloc, calloc. Когда программа вызывает одну из этих функций, операционная система выделяет участок памяти нужного размера. Эта память может быть неинициализированной (malloc) или проинициализированной нулями (calloc).

Освобождение памяти:

Для освобождения динамически выделенной памяти используется функция free. Когда программа вызывает free для указателя, который ранее был возвращен malloc или calloc, операционная система помечает соответствующий блок памяти как свободный.