**Лабораторная работа №1**

**УКАЗАТЕЛИ И СПИСКИ**

Цель работы: научиться строить однонаправленные, двунаправленные и кольцевые списки на базе указателей.

Порядок выполнения работы

1. Ознакомиться с теоретической частью лабораторной работы.

2. Реализовать практическое задание.

3. Оформить отчет по лабораторной работе..

Многочлен image002.gif с целыми коэффициентами можно представить в виде списка. При этом, если image004.gif то соответствующий элемент не включается в список. На рис. 1 показано общее представление многочлена и пример image006.gif

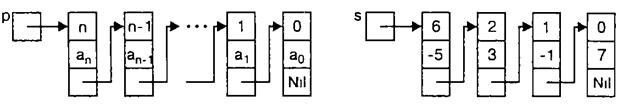


Рис. 1 – Представление многочленов с помощью списков

Необходимо описать тип данных, соответствующий предложенному представлению многочленов, а также разработать следующие функции и процедуры для работы с этими списками-многочленами:

- логическую функцию ***Equality(p,q)***, проверяющую равенство многочленов p и q;

- функцию ***Meaning(p, x)***, вычисляющую значение многочлена в целочисленной точке х;

- процедуру ***Add(p,q,r)*** вычисления суммы многочленов q и r, результат – многочлен p.

*1.*

*2.*

*3.*

*4.*

*5.*

*Контрольные вопросы:*

1. Перечислите наиболее частые случаи, когда может оказаться эффективной работа с указателями.

2. Объясните, в чем различие между типизированными и нетипизированными указателями.

3. Объясните, почему сложение указателей не имеет смысла.

4. Дайте определение абстрактному типу данных «список».

**Лабораторная работа №2**

**ПОСТРОЕНИЕ СЛОВАРЕЙ НА ОСНОВЕ МЕТОДА ОТКРЫТОГО ХЕШИРОВАНИЯ ДАННЫХ**

Цель работы: научиться строить словари на базе линейных списков и открытого хеширования данных.

Порядок выполнения работы

1. Ознакомиться с теоретической частью лабораторной работы.

2. Реализовать практическое задание.

3. Оформить отчет по лабораторной работе.

*Задание.* На основе динамических списков необходимо реализовать словарь. Основные операции, выполняемые над данными словаря:

- поиск,

- вставка;

- удаление.

В сочетании со списками для построения словарей удобно использовать *открытое хеширование данных*, позволяющее фиксировать время выполнения операторов над словарем, а также сделать *потенциально бесконечным пространство* для хранения данных.

На рис. 1 показана базовая структура данных при открытом хешировании. Основная идея метода заключается в том, что множество данных (возможно, очень большое) разбивается на конечное число классов. Для В классов, пронумерованных от 0 до В-1, строится хеш-функция h такая, что для любого элемента x исходного множества функция h(x) принимает целочисленное значение из интервала 0, …, В-1, которое соответствует классу, которому принадлежит элемент x. Элемент x называют ключом, h(x) – хеш-значением х, а классы – сегментами. Массив (таблица сегментов), проиндексированный номерами сегментов 0, 1, … В-1, содержит заголовки для В списков. Элемент х i-го списка – это элемент исходного множества, для которого h(x)=i.

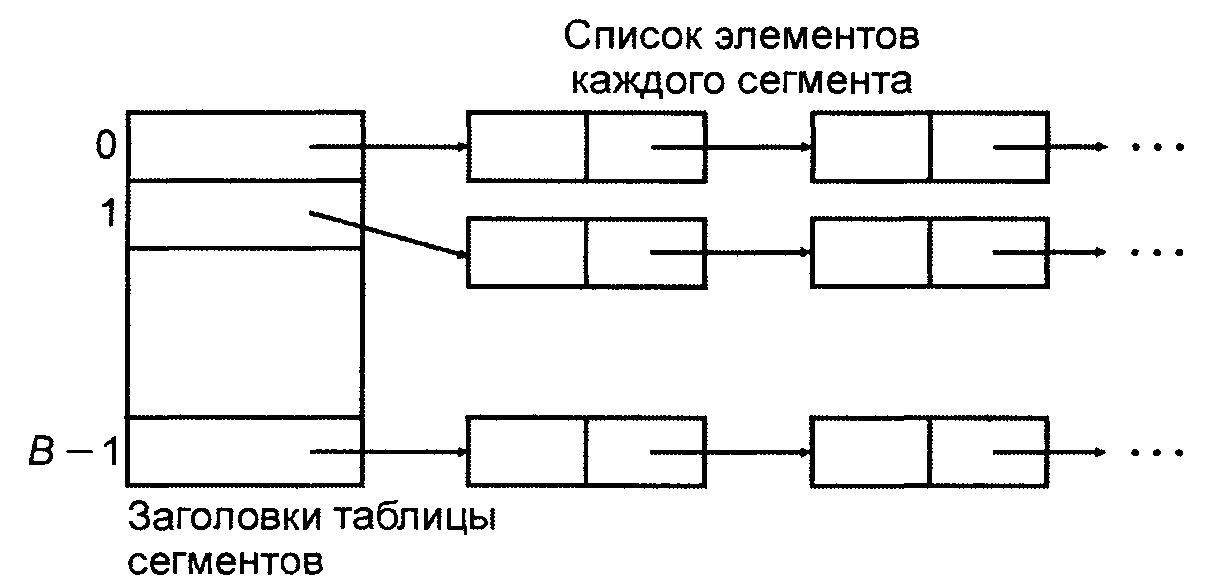


Рис. 1 – Организация данных при открытом хешировании

Если сегменты приблизительно равны по размеру, то в этом случае списки всех сегментов должны быть наиболее короткими при данном числе сегментов. Если исходное множество состоит из N элементов, тогда средняя длина списков будет N/B элементов. Если удается оценить величину N и выбрать B как можно ближе к этой величине, то в каждом списке будет один-два элемента. Тогда время выполнения операций с данными будет малой постоянной величиной, зависящей от N или от В. Однако не всегда ясно, как выбрать хеш-функцию h так, чтобы она примерно поровну распределяла элементы исходного множества по всем сегментам.

Идеальной хеш-функцией является такая, которая для любых двух неодинаковых ключей выдает неодинаковые адреса, т.е.

r:\Temp\Rar$DRa0.289\ЭУМКД_СиАОД\Практика\content\lb2\lb2.files\image004.gif

Однако подобрать такую функцию можно в случае, если все возможные значения ключей известны заранее. Такая организация данных носит название «совершенное хеширование». Если заранее не определено множество значений ключей, и длина таблицы ограничена, подбор совершенной функции затруднителен. Поэтому часто используют хеш-функции, которые не гарантируют выполнение условия (1).

*Контрольные вопросы:*

1. С какой целью выполняется хеширование данных?

2. Какая хеш-функция является идеальной?

3. В чем отличия между открытым и закрытым хешированием данных?

4. Объясните суть коллизий при закрытом хешировании данных.

**Лабораторная работа №3**

**ПОСТРОЕНИЕ ПРИОРИТЕТНЫХ ОЧЕРЕДЕЙ И ОБРАБОТКА ДАННЫХ НА ИХ ОСНОВЕ**

Цель работы: научиться строить словари на базе линейных списков и открытого хеширования данных.

Порядок выполнения работы

1. Ознакомиться с теоретической частью лабораторной работы.

2. Реализовать практическое задание.

3. Оформить отчет по лабораторной работе.

Название *«очередь с приоритетом»* предполагает, что объекты, требующие обработки, ставятся в очередь, а извлекаются из нее не в порядке занесения, а согласно приоритету.

Алгоритмы приоритетного обслуживания очень популярны во многих областях вычислительной техники, в частности в ОС, когда одним приложениям нужно отдать предпочтение перед другими при их обработке в мультипрограммной смеси. Весь трафик разбивается на небольшое количество классов, каждому из которых присваивается приоритет. Приоритетное обслуживание обычно применяется для класса трафика, чувствительного к задержкам, имеющего небольшую интенсивность. Тогда обслуживание этого класса не слишком ущемляет остальные классы. Например, голосовой трафик (чувствителен, но его интенсивность обычно не превышает 8-16 Кбит/c).

Альтернативой приоритетному обслуживанию являются взвешенные очереди. Они гарантируют всем классам трафика определенный минимум пропускной способности. Под весом понимается процент предоставляемой классу трафика пропускной способности от полной пропускной способности выходного интерфейса. С каждой очередью связывается процент пропускной способности ресурса, гарантируемый ему при перегрузках этого ресурса. Совмещение достоинств приоритетных и взвешенных очередей удается получить в комбинированных алгоритмах. Обычно в них используется одна приоритетная очередь для чувствительного трафика, а остальные обслуживаются в соответствии с взвешенным алгоритмом. Им выделяется часть интенсивности ресурса, оставшегося от приоритетной очереди.

*Задание.*

Необходимо реализовать очередь на базе списков, применяя комбинированный алгоритм для ее обслуживания. Затем продемонстрировать выполнение основных операций с элементами очереди: поиск, добавление, удаление.

Контрольные вопросы и задания:

1. Дайте определение абстрактному типу данных «очередь».

2. Перечислите основные операторы, которые определены для работы с очередью.

3. Дайте определение многопоточной очереди.

4. Назовите особенности циклической очереди и приведите пример способа ее организации.

**Лабораторная работа №4**

**Использование стеков для построения различных форм представления выражений**

Цель работы: научиться строить префиксную, постфиксную и инфиксную формы выражения, а также преобразовывать выражения из одной формы в другую.

**Порядок выполнения работы**

1. Ознакомиться с теоретической частью лабораторной работы.

2. Реализовать практическое задание.

1. Оформить отчет  по лабораторной работе.

Большую часть задач, решаемых с помощью программирования, составляют задачи, в которых широко применяются методы вычислительной математики, а в них входят арифметические и логические выражения. Наиболее популярны три формы записи выражений, отличающиеся друг от друга взаимным расположением операторов и операндов.

* **А+В** – инфиксная: знак операции находится между операндами;
* **+АВ**  – префиксная (польская): знак операции расположен перед операндами;
* **АВ+ –** постфиксная (обратная польская): знак операция находится после операндов.

Хотя префиксная и постфиксная формы записи, на первый взгляд, кажутся не очень наглядны, они чаще инфиксной используются в вычислительной технике для обработки выражений.

Для преобразования выражений из инфиксной в постфиксную и префиксную формы нужно учитывать правила приоритетности операций. Операции с высшим приоритетом преобразуются первыми, а после преобразования операция рассматривается как один операнд. Общепринятую приоритетность операций можно изменить при помощи скобок. При просмотре строки, не содержащей скобок, вычисления выполняются слева направо для операций с одинаковым приоритетом, за исключением случая возведения в степень, когда вычисления выполняются справа налево.

Особенность обратной польской записи состоит в следующем: в ней отсутствуют скобки, операнды располагаются в том же порядке, что в исходном выражении, а знаки операций при просмотре записи слева направо встречаются в том порядке, в котором нужно выполнять соответствующие действия. Отсюда вытекает основное преимущество обратной польской записи перед обычной записью выражений со скобками: *выражение можно вычислить в процессе однократного просмотра слева направо.*

Алгебраическое преобразование инфиксного выражения в обратное польское или польское основано на приоритетах операторов и предлагает использование стека. Обратное польское (польское) выражение хранится в виде выходной строки, используемой в дальнейшем при генерации объектного кода. В ходе преобразования инфиксного выражения в обратное польское порядок всех переменных и констант не меняется, а порядок операторов выходной строки соответствует их приоритетам.

Алгоритм преобразования выражения из инфиксной формы в префиксную запись рассмотрим на примере выражения a+b/(c-d).

1. Необходимо переписать выражение справа налево: (d-c)/b+a;
2. Воспользовавшись алгоритмом постфиксной трансляции, получим: dc-b/a+;
3. Полученную строку требуется записать справа налево, в результате чего получается выражение в префиксном виде: +a/b-cd.

*Задание.*

1. Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в постфиксную форму представления.
2. Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в префиксную форму представления.

Для обоих алгоритмов предусмотреть вхождение операций с различными приоритетами, а также наличие скобок в инфиксных выражениях.

*Контрольные вопросы:*

1. Дайте определение абстрактному типу данных «стек»;
2. Перечислите основные операторы, которые определены для работы со стеком;
3. Назовите основное преимущество обратной польской записи перед обычной записью выражений со скобками;
4. Каким образом используется стек для преобразования выражений из одной формы записи в другую?