МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Курсовая работа
по курсу "Вычислительные системы"
2 семестр

Задание 8
Линейные списки

Автор работы:
студент 1 курса, группы М8О-106Б-20
Леухин М. В.
Преподаватель:
Дубини А. В.

Москва, 2021

Содержание

1 Постановка задачи

Задание: Составить и отладить программу на языке Си для обработки линейного списказаданной организации с отображением списка на массив (только с индексным доступом, без применения указателей). Навигацию по списку следует реализовать с применением итераторов. Предусмотреть выполнение одного нестандартного и четырёх стандартных действий:

1. Печать списка.
2. Вставка нового элемента в список.
3. Удаление элемента из списка.
4. Подсчёт длины списка.

Тип элементоа списка: вещественный.
Вид списка: линейный двунаправленный с браьерным элементом.
Нестандартное действие: проверить упорядоченность элементов списка.

2 Основная часть

2.1 Линейный список

Линейный список — базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Линейные списки являются обобщением последовательных структур с ограниченным доступом: файлов, очередей и стеков. Они позволяют нам представить элементы так, чтобы в отличие от стека и очереди, каждый элемент был бы доступен и для этого не нужно было бы извлекать из структуры предшествующие элементы. Линейным списком можно назвать представление в ЭВМ конечного упорядоченного множества элементов одного типа. Точнее будет сказать не множества, а мультимножества — в последовательности могут находиться элементы с одинаковым значением. Элементы этого множества линейно упорядочены, но порядок определяется не номерами (индексами, как в массиве), а относительным расположением элементов. Отношений порядка на этом множестве может быть не одно, а два, и тогда мы имеем дело с двусвязным списокм. Доопределив в качестве следующего за последним первый элемент списка, поулчим кольцевой линейный список. Линейный список обозначается простым перечислением элементов в заданном порядке в круглых скобках через запятую, например:

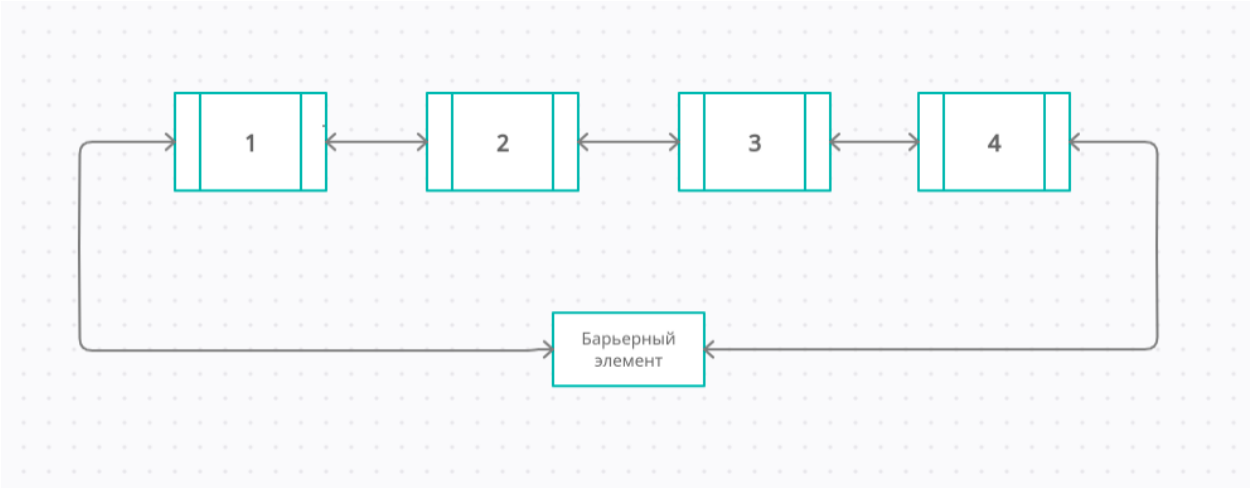
$$l = (t_1, t_2, t_3, t_4, t_5)$$

Линейные списки естественно использовать всякий раз, когда встречаются упорядоченные множества переменного размера, где включение, поиск и удаление элементов должны выполняться не систематически в голове или хвосте, как для файлов или стеков, а в произвольных последовательно достигаемых местах, но с сохранением порядка следования остальных элементов. Таким порядком могли бы быть, например, приоритеты, присвоенные заданиям, ожидающим обработки в операционной системе или распечатки в сетевом принтере.

Заметим, что порядок следования элементов в списке не предполагает упорядоченности хранимых в этих элементах значений.

Пару слов о линейном двунаправленном списке с браьерным элементом — элементы такого списка имеют указатель как на следующий элемент, так и на предыдущий. Наличие барьерного элемента означает, что такой список никогда не бывает пуст — барьерный элемент является предыдущим для первого элемента и следующим для последнего; пустой список будет состоять из единственного элемента — браьерного. Ниже представлено схематическое изображение данного списка.

Среди возможных реализаций линейного списка была выбрана реализация на массиве. Преимуществом такой реализации является то, что в случае массива (в отличие от динамическим структур) гарантируется последовательное хранение элементов массива в памяти компьютера, что уменьшает время доступа к элементам (при использоваине динамических структур имеем дополнительную нагрузку на систему при переходе по указателям к различным местам памяти).



2.2 Итераторы

Для удобства организации списка и обеспечения единообразия доступа к нему определим объекты, обладающие функциями перехода от данного элемента списка к соседним. Зададим для них отношения равенства и неравенства — два таких объекта равны тогда и только тогда, когда они указывают на один и тот же элемент одного спсика. Также предоставим возможность чтения и записи элемента спсика посредством введённых объектов. Такие объекты принято называть *итераторами*.

Итератор — это интерфейс, предоставляющий доступ к элементам структуры и навигацию по ним.

2.3 Описание алгоритма и оценка сложности

Цель: написать функцию, выполняющую потавленную задачу — проверка упорядоченности элементов списка.

Сформулируем задачу более точно — проверить упорядоченность значений элементов списка. Сразу же из формулировки можем определить сложность алгоритма — мы не можем узнать, упорядоченны ли элементы по значению, не просмотрев все элементы, а потому имеем сложность $O(n)$.

Как будет выглядеть сама функция? Так как в задании не указано конкретно, как должны быть упорядочены значения, необходимо рассматривать общий случай — элементы могут находиться как в неубывающей, так и в невозрастающей последовательности. В начале работы функции возьмём 2 первых элемента списка и запишем их в условные переменные *left* и *right*. Если они окажутся равны по значению, будем в переменную *right* записывать значение следующего элемента до тех пор, пока не окажется $left \neq right$ (при этом учитываем, что на каждой итерации список может закончиться — в таком случае все элементы списка равны и список отсортирован; частный случай — список из одного элемента всегда отсортирован). Как только мы нашли такой *right*, что $left \neq right$, можем определить тип последовательности — неубывающая или невозрсатющая. После этого в цикле проходим по всему списку и рассматрвиаем 2 смежных элемента. Если их упорядоченность не совпадает с упорядоченностью всех предшествующих пар списка — список не отсортирован, завершаем выполнение функции. Если же

на очередной итерации мы встречаем конец списка — список отсортирован, завершаем выполнение функции.

3 Вывод

Линейный список — очень мощный инструмент для работы с упорядоченным набором значений в том случае, когда часто приходиться осуществлять вставку или удаление элементов не с концов последовательности. А релазиация на массиве позволяет увеличить эффективность программы, так как хранение элементов списка в массиве гарантирует хранение этих элементов в одной области памяти. Особенно эффективна будет такая реализация в том случае, если мы знаем, что количество элементов списка никогда не превысит определённого числа — тогда элементы можно хранить в статическом массиве и не затрачивать ресурсы ЭВМ на перераспределение памяти.

Итератеры — полезные инструменты, обеспечивающие интерфейс для работы со структурой. Если впоследствии будет принято решение изменить структуру списка, достаточно будет лишь переписать функции, связанные с итератерами — их работа закреплена функциональной спецификаицей (то есть гарантируется, что каждая функция принимает определённый набор значений и возвращает определённый результат), что не приведёт к нарушению работы уже написанного кода.

4 Список источников

- Методические указания к выполнению курсовых работ. Зайцев В. Е.
- Курс информатики. Гайсарян С. С., Зайцев В. Е.