Дмитриева Марина Валерьевна

ОСНОВНЫЕ АЛГОРИТМЫ РАБОТЫ С ЛИНЕЙНЫМИ ЦЕПНЫМИ СПИСКАМИ

При создании любой сложной структуры данных надо уметь решать задачи поиска, то есть определять, принадлежит ли некоторый элемент построенному множеству элементов. Применительно к списку задача поиска конкретизируется следующим образом: требуется определить, имеется ли в списке элемент, информационное поле которого совпадает с заданным значением (образцом).

Возможны два варианта ответа на вопрос, найден ли элемент в списке. В первом случае можно выдавать логическое значение **true**, если элемент найден, и **false** в противном случае. Во втором варианте в качестве ответа может быть ссылка на найденный элемент или значение **Nil**, если элемента в списке не обнаружено.

...надо уметь решать задачи поиска...

При описании процедур работы со списками будем считать, что определены структуры данных, представленные в листинге 1.

ПОИСК ЭЛЕМЕНТА В СПИСКЕ. ВАРИАНТ 1

Пусть задан линейный список с указателем **t** на первый элемент. Требуется определить, есть ли в списке элемент с заданным значением **r** информационного поля.

Указатель **cur** устанавливается на текущий элемент, в начальный элемент он совпадает с **t**. Для поиска элемента в линейном списке поступаем следующим образом. Сравниваем значение информационного поля текущего элемента со значением **r**. Если значения совпадают, то задача решена, в противном случае требуется перейти к следующему элементу списка.

Процесс просмотра элементов списка продолжается до тех пор, пока не будет найден соответствующий элемент, либо пока не исчерпается весь список. В последнем случае нужного элемента в списке нет.

Листинг 1. Описание структур данных для работы с линейными списками

```
type
   elem_list = integer; {тип информационного поля элемента списка}
   list = ^elem; (указатель на элемент списка)
   elem= record info: elem_list; (информационное поле элемента списка)
                next: list {указатель на следующий элемент}
         end;
```

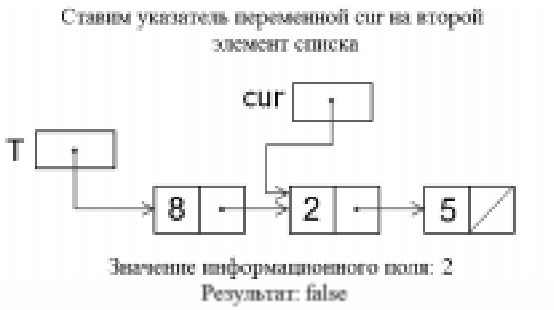


Рис. 1. Поиск элемента в списке

Процесс просмотра элементов списка...

Функция, осуществляющая поиск элемента в линейном списке, приведена в листинге 2. Функция выдает значение **true**, если элемент найден, и **false** в противном случае.

На рис. 1 изображена ситуация в точке {1}, которую можно описать так: просмотрен линейный список до элемента, предшествующего элементу, на который установлен указатель **cur**. Среди просмотренных элементов нет элемента с заданным информационным полем (равного образцу).

Указатель **cur** установлен на первый элемент непросмотренной части списка. В точке {1} верно и следующее утверждение: не просмотренная часть списка не пуста. Точка {2} соответствует тому, что цикл завершил свою работу, значение переменной **cur** равно **nil**, список просмотрен полностью, элемент не найден.

ПОИСК ЭЛЕМЕНТА В СПИСКЕ. ВАРИАНТ 2

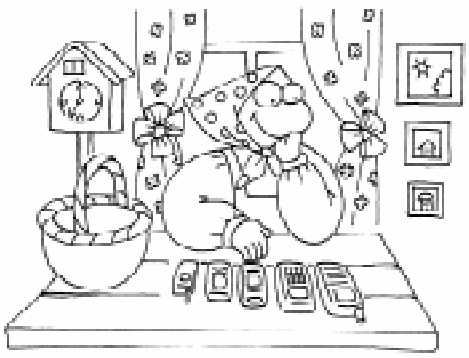
Приведем еще один вариант описания функции поиска элемента в линейном списке, использующий особенности структур данных языка Паскаль. Описание функции приведено в листинге 3.

Листинг 2. Поиск элемента в списке. Вариант 1

```
function search1 (var t: list; r: elem_list): boolean;
  var cur: list;
begin search1 := false;
      cur := t;
      while cur <> nil do {1}
        if cur^.info = r
        then
           begin search1 := true;  exit end
        else cur := cur^.next {2}
end
```

Листинг 3. Поиск элемента в списке. Вариант 2

```
Function search2 (var t: list; r: elem_list): boolean;
  var cur: list; sel: Boolean;
begin sel := false;
      cur := t;
      while sel < (cur <> nil) do
        begin sel := cur^.info = r;
              cur := cur^.next
        end;
       search2 := sel
end
```



...алгоритм сортировки линейного списка в порядке возрастания...

Напомним, что стандартный тип **Boolean** считается перечислимым типом, определенным следующим образом: **type Boolean = (false, true)**. Так как элементы в перечислимом типе считаются упорядоченными, то верно соотношение **false < true**. Тело цикла выполняется, если истинно выражение **sel < (cur <> nil)**.

Если в списке найден элемент с информационным полем **r**, то значение логической переменной **sel** станет равным **true** в результате выполнения оператора присваивания **sel:=cur^.info=r**. Так как значение выражения **true < L** ложно при любом значении логической переменной **L**, то выражение, управляющее работой цикла **sel < (cur <> nil)** также станет равным **false**, и цикл завершит свою работу.

Если же просмотрен весь список и в нем нет элемента с информационным полем **r**, то значение переменной **sel** равно **false**, значение выражения **cur<>nil** после просмотра всего списка равно **false**. Формула **false < false** имеет значение **false**, поэтому работа цикла будет завершена. Результат поиска совпадает со значением переменной **sel**.

СОРТИРОВКА СПИСКА

Предположим, что на элементах, из которых строится список, задано отношение порядка. Рассмотрим алгоритм сортировки линейного списка в порядке возрастания значений информационных полей его элементов.

Пусть список задан указателем на первый элемент **t**. Будем использовать два указателя **p** и **q**, указатель **p** устанавливается на очередной элемент списка, указатель **q** пробегает значения от элемента, непосредственно следующего за тем, на который установлен указатель **p**, до конца списка.

Сравниваются информационные поля элементов (**s1** и **s2**), на которые установлены указатели **p** и **q**. Если верно, что **s1 > s2**, то информационные поля меня-

Листинг 4. Сортировка списка в порядке возрастания информационных полей

```
procedure sort_list  (var t: list);
  var p,q: list; r: elem_list;
begin p:=t;
      while p <> nil do
        begin
          q:= p^.next;
          while q <> nil do
            begin if p^.info > q^.info
                  then
                     begin
                       r := p^.info;
                       p^.info := q^.info;
                       q^.info :=r
                      end;
                 q := q^.next
            end;
          p := p^.next {1}
        end
end
```

Рис. 2. Сортировка элементов списка в порядке возрастания

ются местами. После того, как для фиксированного **p** переменная **q** пробежит значения до конца списка, элемент, на который указывает **p**, имеет информационное поле, наименьшее из всех информационных полей следующих за ним элементов.

Дальнейшие действия состоят в том, чтобы перейти к следующему за **p** (если он есть) элементу списка и для него повторить: установить указатель **q** на следующий за **p** элемент и сравнить информационные поля элемента, на который установлен указатель **p**, и всех следующих за ним элементов.

Если просмотрен весь список, то анализ списка завершен и элементы списка упорядочены. До начала просмотра списка значение **p** совпадает с **t**, указатель **q** установлен на второй элемент списка, если список содержит более одного элемента. Процедура, осуществляющая сортировку элементов списка в порядке возрастания информационных полей, приведена в листинге 4.

На рис. 2 представлена ситуация, когда список, предшествующий элементу, на который установлен указатель **p**, уже отсортирован, указатель **q** установлен на элемент, непосредственно следующий за **p**. После того как указатель **q** «пробежит» по всем элементам «хвоста» списка, элемент, на который установлен указатель **p**, будет иметь наименьшее значение информационного поля из всех элементов «хвоста» списка.

После перехода к следующему элементу списка в точке {1} просмотренная часть списка (начало списка) отсортирована. Значение **p**, равное **nil**, будет означать, что просмотрен весь список и его элементы расположены в порядке возрастания.

СЛИЯНИЕ ДВУХ УПОРЯДОЧЕННЫХ СПИСКОВ

Рассмотрим два упорядоченных списка. Задача слияния двух упорядоченных списков состоит в формировании упорядоченного списка, состоящего из тех же элементов, что и исходные. Результирующий список должен быть упорядоченным.

Для решения задачи будем параллельно просматривать два списка с указателями **t** и **p**. Пусть **t1** – указатель на первый элемент непросмотренной части первого списка, указатель **p1** устанавливается на текущий элемент второго списка. В начальный момент времени оба указателя установлены на первые элементы списков (если списки не пусты).

Все просмотренные элементы хранятся в списке с указателем на первый элемент **s**, указатель **s1** установлен на последний элемент построенного (уже слитого) списка. Естественно, сначала формируемый список пуст. Сравниваем информационные поля

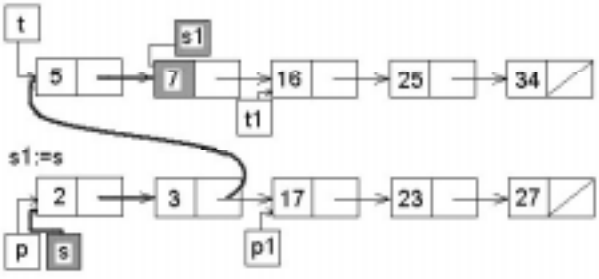


Рис. 3. Слияние двух упорядоченных списков

Задача слияния двух упорядоченных списков...

Листинг 5. Процедура слияния двух упорядоченных списков

```
procedure sl_list  (var t, p, s: list);
  var t1, p1, s1: list;
begin if t = nil then s := p  else if p= nil then s := t
        else {оба списка не пусты}
        begin
           t1 := t; p1 := p;
           if t1^.info < p1^.info
           then   begin s := t1; t1 := t1^.next end
           else   begin s := p1; p1 := p1^.next end;
           s1:=s;
           while (t1 <> nil) and (p1 <> nil) do
             begin if t1^.info < p1^.info
                   then  begin s1^.next :=t1; t1 := t1^.next end
                   else  begin s1^.next :=p1; p1 := p1^.next end;
                   s1:=s1^.next
              end
         end
end
```

элементов, на которые установлены указатели **t1** и **p1**.

Тот элемент, информационное поле которого меньше, добавляется в конец формируемого списка, а указатель в соответствующем списке сдвигается на следующий элемент списка.

Просмотр списков следует продолжать до тех пор, пока не исчерпается один из списков. Процедура слияния двух упорядоченных списков приведена в листинге 5.

На рис. 3 изображена ситуация, когда просмотрены элементы первого списка до элемента с указателем **t1**, просмотрены элементы второго списка до элемента с указателем **p1**, просмотренные части обоих списков «слиты», указатель **s** установлен на первый элемент списка-результата, указатель **s1** установлен на последний элемент формируемого списка.

Просмотр списков продолжается, пока не исчерпается один из них. Не просмотренная часть другого списка составляет «хвост» списка результата.

ЗАДАЧИ

1. Опишите функцию, которая определяет число элементов в списке, информационные поля которых совпадают с заданным значением.

2. Опишите функцию, которая проверяет, имеются ли в списке элементы, непосредственно следующие друг за другом, информационные поля которых одинаковы.

3. Опишите функцию, которая определяет, образуют ли элементы списка возрастающую последовательность.

4. Опишите следующую модификацию рассмотренной процедуры сортировки линейного списка. При каждом фиксированном значении **p** определите указатель на элемент с минимальным значением информационного поля для элементов, следующих за **P**, и лишь затем в случае необходимости меняйте значения информационных полей.

5. Опишите процедуру слияния двух упорядоченных списков, исходные списки должны остаться без изменения.

Дмитриева Марина Валерьевна, доцент кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета.