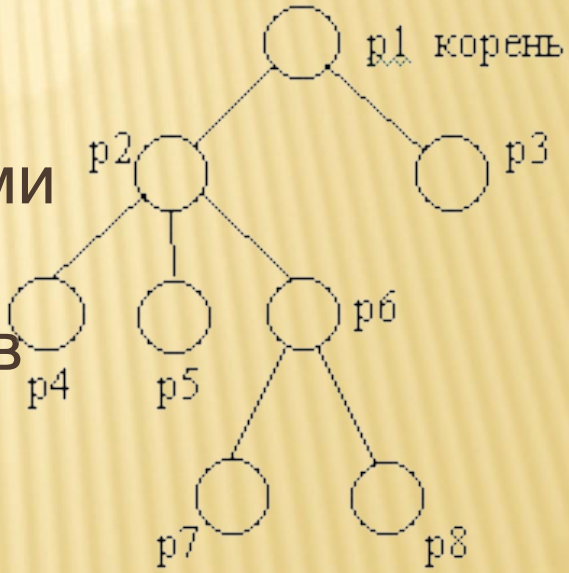
БИНАРНЫЕ ДЕРЕВЬЯ.

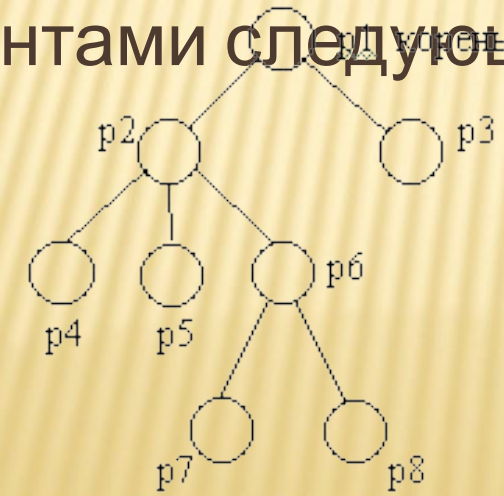
ОСНОВНЫЕ ОПЕРАЦИИ С БИНАРНЫМИ ДЕРЕВЬЯМИ.

ДЕРЕВО - СТРУКТУРА, КОТОРАЯ ХАРАКТЕРИЗУЕТСЯ СЛЕДУЮЩИМИ СВОЙСТВАМИ:

1. существует единственный элемент, на который не ссылается никакой другой элемент. Этот элемент называется **корнем**;
2. каждый элемент связан с несколькими элементами следующего уровня иерархии. Эти элементы могут быть в свою очередь деревьями (поддеревьями);
3. каждый элемент промежуточного уровня порожден только одним элементом более высокого уровня.



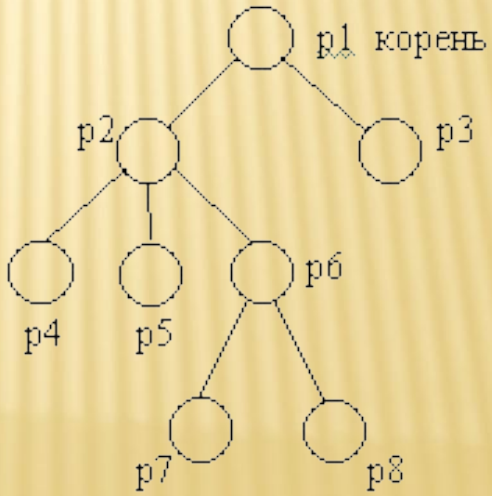
- Элементы дерева, которые не ссылаются на другие элементы, являются **терминальными** (т.е. конечными) или **листьями**. А элементы, не являющиеся терминальными, называются **внутренними узлами**.
- Таким образом, дерево отражает иерархически упорядоченную структуру данных, в которой прослеживаются связи между элементами предыдущего (верхнего) уровня или **предками** и элементами следующего уровня – **потомками**.



p1, p2, p3, p4, p5, p6, p7, p8 - узлы

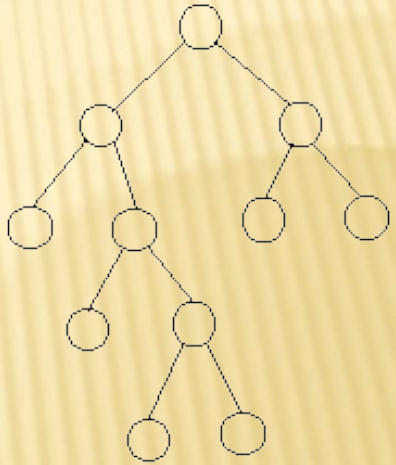
p3, p4, p5, p7, p8 - листья

- Узлы располагаются по уровням.
- **Корень** – нулевой уровень и т.д.
- Максимальный уровень какого-либо элемента дерева называется его **глубиной** или **высотой**.
- Число непосредственных потомков внутреннего узла называется его **степенью**.
- Максимальная степень всех узлов есть **степень дерева**.

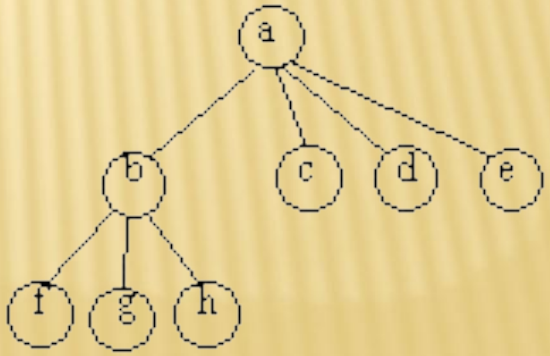


ДВА ТИПА ДЕРЕВЬЕВ: **БИНАРНЫЕ** И **СИЛЬНОВЕТВЯЩИЕСЯ**.

- В **бинарном** дереве из каждой вершины выходит не более двух дуг.



- В **сильноветвящемся** дереве количество дуг может быть произвольным.

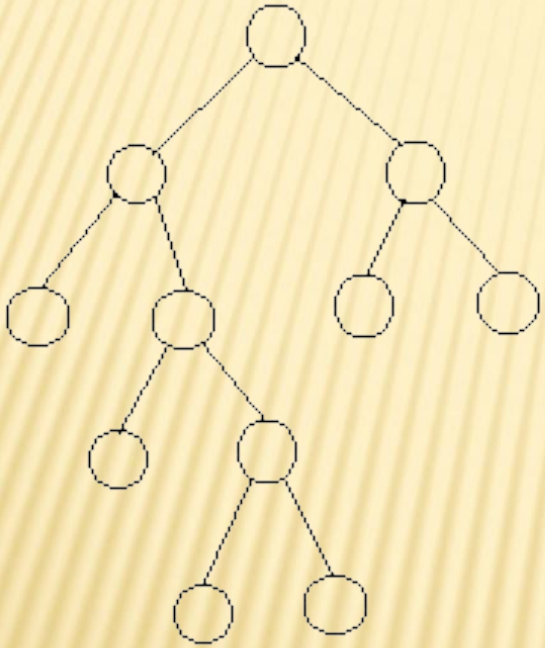


ОСНОВНЫЕ ОПЕРАЦИИ НАД ДЕРЕВЬЯМИ:

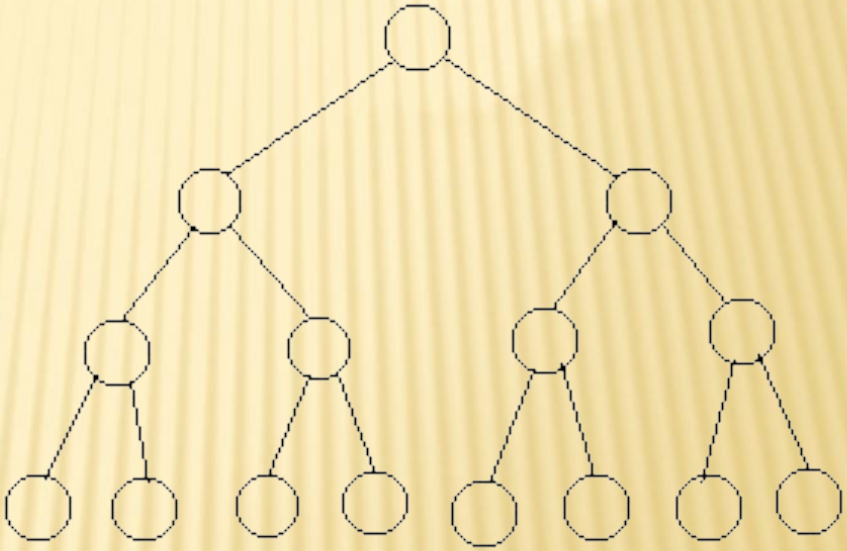
- пройти все узлы в определенном порядке,
- найти узел с заданным свойством,
- определить отца данного узла,
- определить сыновей данного узла,
- удалить определенный узел (поддерево),
- добавить новый узел
- и т.д.

ПОЛНОЕ И НЕПОЛНОЕ БИНАРНЫЕ ДЕРЕВЬЯ

а) неполное бинарное дерево



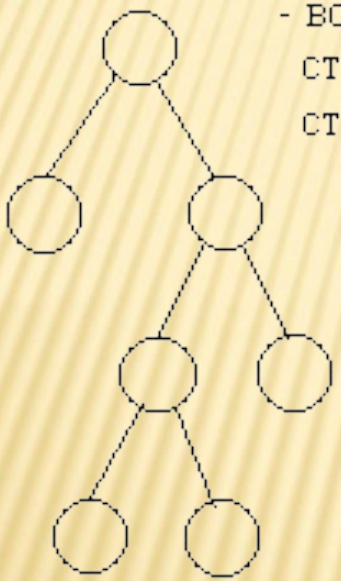
б) полное бинарное дерево



- на всех уровнях меньше, чем n, узлы имеют степень 2, на уровне n - 0

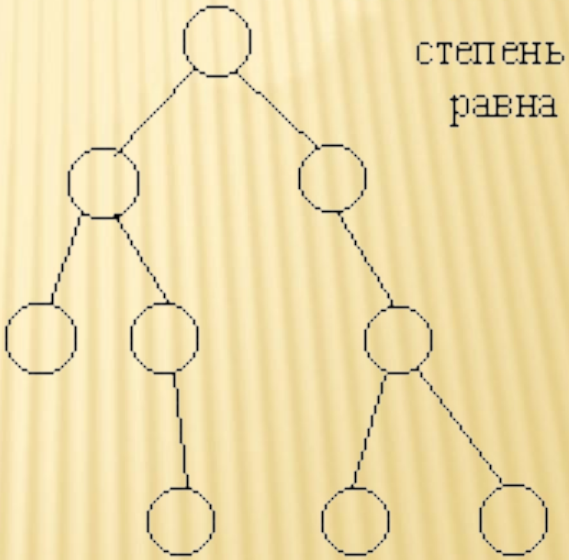
СТРОГО И НЕ СТРОГО БИНАРНЫЕ ДЕРЕВЬЯ

а) строго бинарное дерево



- все узлы имеют степень 2 или степень 0

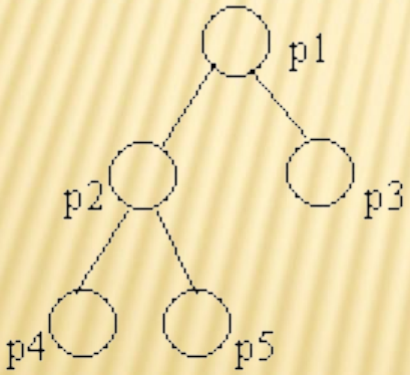
б) не строго бинарное дерево

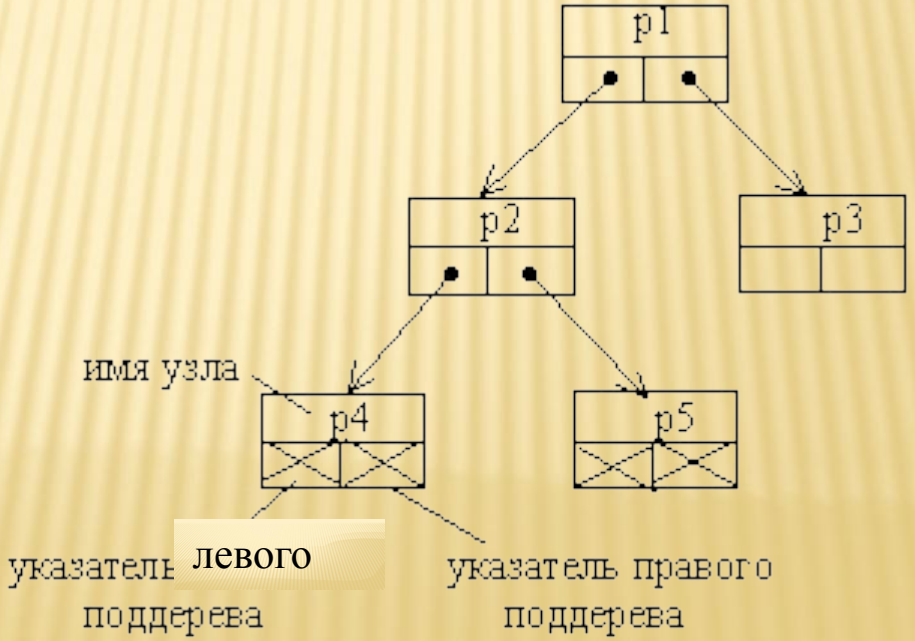


степень узлов равна 2, 1, 0

ПРЕДСТАВЛЕНИЕ БИНАРНЫХ ДЕРЕВЬЕВ :

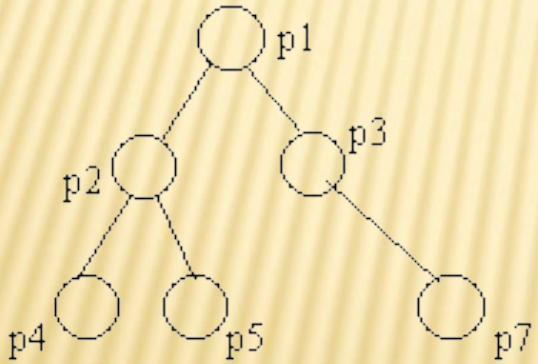
- **Списочное представление** бинарных деревьев основано на элементах, соответствующих узлам дерева.
- Каждый элемент имеет **поле данных** и **два поля указателей.** Один указатель используется для связывания элемента с правым потомком, а другой - с левым. Листья имеют пустые указатели потомков.





ПРЕДСТАВЛЕНИЕ БИНАРНЫХ ДЕРЕВЬЕВ :

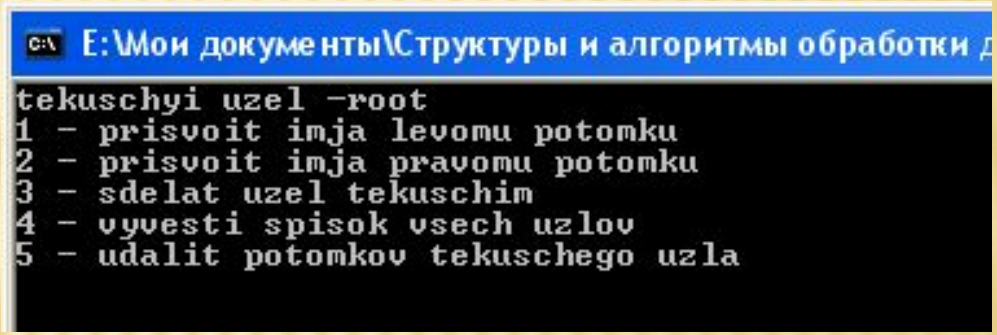
- В виде массива проще всего представляется полное бинарное дерево, так как оно всегда имеет строго определенное число вершин на каждом уровне. Вершины можно пронумеровать слева направо последовательно по уровням и использовать эти номера в качестве индексов в одномерном массиве.



ПРИМЕР:

Разработать программу создания и редактирования бинарного дерева:

1. Добавление узлов
2. Удаление узлов
3. Задание текущего узла
4. Отображение на экране дерева



```
program bin_tree_edit;

type node=record
            name: string;
            left, right: pointer;
        end;
var
        n:integer;
        pnt_s,current_s,root: pointer;
        pnt,current:^node;
        s: string;
```

{Поиск узла по содержимому}

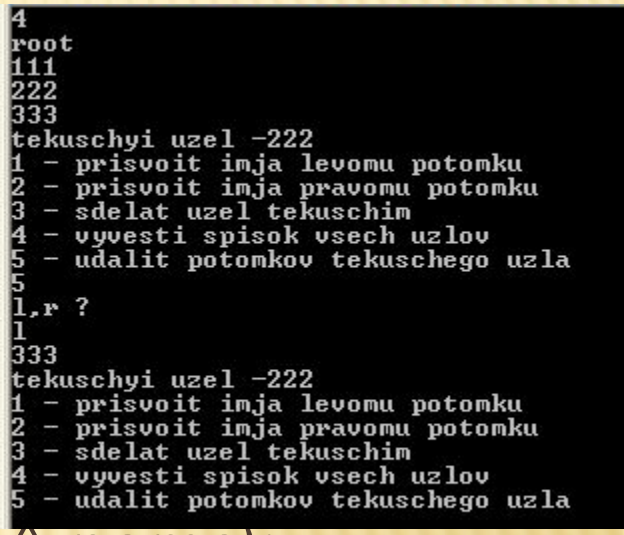
```
procedure node_search (pnt_s:pointer; var current_s:pointer)
var
        pnt_n:^node;
begin
pnt_n:=pnt_s; writeln(pnt_n^.name);
if not (pnt_n^.name=s) then
        begin
                if pnt_n^.left <> nil then
                        node_search (pnt_n^.left,current_s);
              if pnt_n^.right <> nil then
                        node_search (pnt_n^.right,current_s);
        end
else current_s:=pnt_n;
End;
```

{Вывод списка всех узлов дерева}

```
procedure node_list (pnt_s:pointer);
var
        pnt_n:^node;
begin
pnt_n:=pnt_s; writeln(pnt_n^.name);
if pnt_n^.left <> nil then node_list (pnt_n^.left);
if pnt_n^.right <> nil then node_list (pnt_n^.right);
end;
procedure node_dispose (pnt_s:pointer);
```

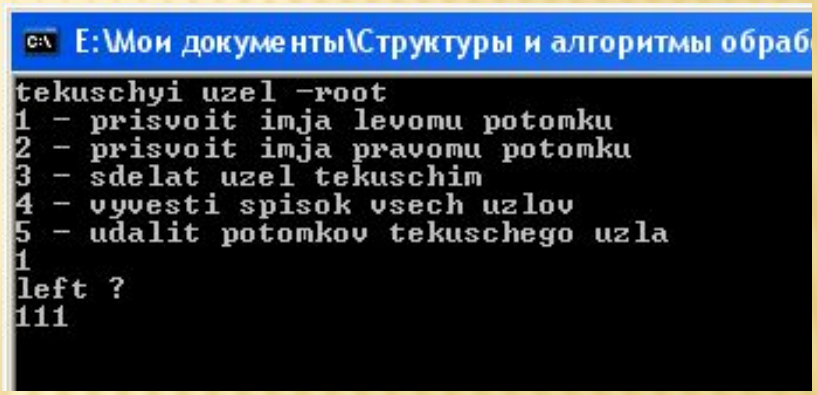
{Удаление узла и всех его потомков в дереве}

```
procedure node_dispose (pnt_s:pointer);
var
        pnt_n:^node;
begin
if pnt_s <> nil then
        begin
                pnt_n:=pnt_s; writeln(pnt_n^.name);
                if pnt_n^.left <> nil then
                        node_dispose (pnt_n^.left);
                if pnt_n^.right <> nil then
                        node_dispose (pnt_n^.right);
                dispose(pnt_n);
        end
End;
```

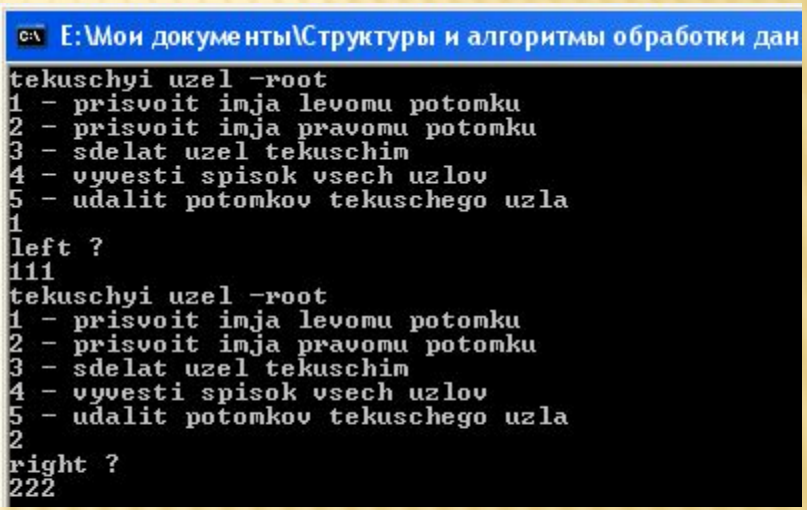


```
{основная программа}
begin
new(current);root:=current;
current^.name:='root';
current^.left:=nil;
current^.right:=nil;
repeat
        writeln('текущий узел -',current^.name);
        writeln('1 – присвоить  имя левому потомку');
        writeln('2 – присвоить  имя правому потомку');
        writeln('3 – сделать  узел текущим');
        writeln('4 – вывести  список всех узлов');
        writeln('5 – удалить  потомков текущего узла');
        read(n);
```

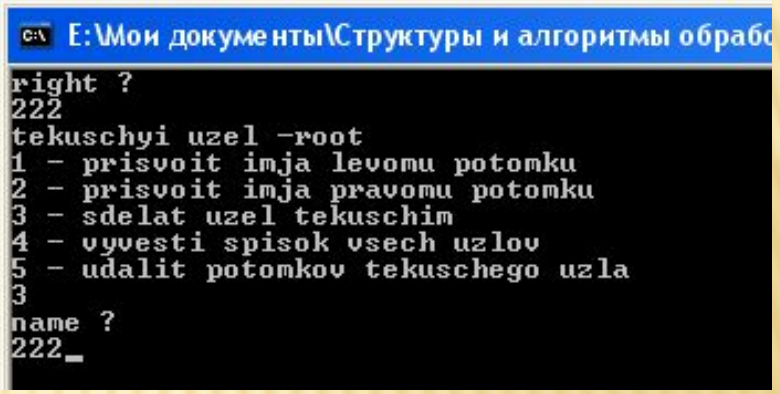
```
if n=1 then
begin {Создание левого потомка}
        if current^.left= nil then new(pnt)
        else pnt:= current^.left;
        writeln('left ?');
        readln;
        read(s);
        pnt^.name:=s;
        pnt^.left:=nil;
        pnt^.right:=nil;
      current^.left:= pnt;
 end;
```



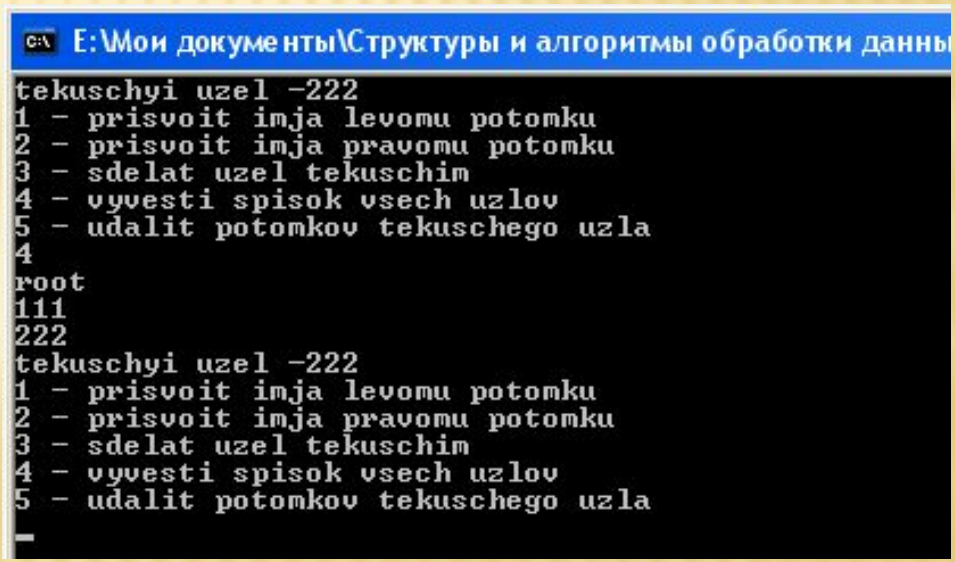
```
if n=2 then
  begin {Создание правого потомка}
        if current^.right= nil then new(pnt)
       else pnt:= current^.right;
       writeln('right ?');
       readln;
       read(s);
       pnt^.name:=s;
       pnt^.left:=nil;
       pnt^.right:=nil;
       current^.right:= pnt;
end;
```



```
if n=3 then
begin {Поиск узла}
        writeln('name ?');
        readln;
        read(s);
        current_s:=nil; pnt_s:=root;
        node_search (pnt_s, current_s);
        if current_s <> nil then
current:=current_s;
end;
```



```
if n=4 then
begin {Вывод списка узлов}
        pnt_s:=root;
        node_list(pnt_s);
end;
```



```
        if n=5 then
        begin {Удаление поддерева}
                writeln('l,r ?');
                readln;
                read(s);
                if (s='l') then
                    begin {Удаление левого поддерева}
                          pnt_s:=current^.left;
                          current^.left:=nil;
                          node_dispose(pnt_s);
                     end
                else
                    begin {Удаление правого поддерева}
                         pnt_s:=current^.right;
                         current^.right:=nil;
                         node_dispose(pnt_s);
                     end;
        end;
until n=0
end.
```

ТЕСТЫ:

Вопрос 1. Какие из указанных ниже структур данных относятся к встроенным:

1) списки;
2) целый тип;
3) дерево;
4) стек.

ТЕСТЫ:

Вопрос 2. Какая из ниже перечисленных структур данных отличается наличием вершины:

1) дерево;
2) множество;
3) стек;
4) массив.

ТЕСТЫ:

Вопрос 3. Описание

```
Var
    i, j : integer;
    x : real;
    s: string;
```

объявляет переменные. Переменная **s** будет является переменной типа:

- целый;
- действительный;
- строка;
- Массив.

ТЕСТЫ:

Вопрос 4. Упорядоченная совокупность элементов некоторого типа, адресуемых при помощи одного или нескольких индексов, называется:

1) массив;
2) дерево;
3) стек;
4) список.

ТЕСТЫ:

Вопрос 5. Структура данных, объединяющая элементы данных разных типов, называется:

1) массив;
2) дерево;
3) стек;
4) запись.

ТЕСТЫ:

Вопрос 6. Структуру данных стек можно организовать с помощью:

1) массивов;
2) деревьев;
3) записей;
4) графов.

ТЕСТЫ:

Вопрос 7. Частным случаем графа является:

- стек;
- очередь;
- дерево;
- матрица.

ТЕСТЫ:

Вопрос 8. В бинарном дереве из каждой вершины выходит:

- произвольное количество дуг;
- не более двух дуг;
- не более трех дуг;
- четное количество дуг.