Определить количество узлов на заданном уровне в бинарном дереве

Описание узла дерева:

```
TNodePtr = ^TNode;
TNode = record
    Data: integer;
    LeftChild, RightChild: TNodePtr;
end;
```

Уровень узла в дереве – это количество его предков. Уровень корня дерева – ноль, уровень его левого и правого сыновей - один, уровень сыновей каждого из них – два, и т.д. В таблице приведено то же дерево, что и в реализации:

| Уровень | Дерево | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 17 | | | | | | | |
| 1 | 11 | | | | 26 | | | |
| 2 | | | 14 | | 21 | | 34 | |
| 3 | | | 12 | 15 | 19 | | | 37 |

Создадим дерево бинарного поиска* и заполним его массивом (17, 11, 14, 12, 15, 26, 21, 19, 34, 37) используя вот такую ~~пороцедуру~~ подпрограмму:

```
procedure Insert(var Node: TNodePtr; ToInsert: Integer);
begin
  if Node = nil then
  begin
    New(Node);
    Node^.Data := ToInsert;
    Node^.LeftChild := nil;
    Node^.RightChild := nil;
  end
  else
    if ToInsert <= Node^.Data then
      Insert(Node^.LeftChild, ToInsert)
    else
      Insert(Node^.RightChild, ToInsert);
end;
```

*Альтернатива – дерево, в котором новый элемент случайно записывается или в левое, или в правое поддерево.

Собственно ~~подпрограмма~~ функция подсчета количества элеменотов на заданном уровне:

```
function CountSiblings(Head: TNodePtr; OnLevel: integer):
                                                    integer;
var
  Count: integer;

  procedure CheckNode(node: TNodePtr; level: integer;
                      var siblCount: integer);
  begin
    if (Node <> nil) then
      if level = OnLevel then
        siblCount := siblCount + 1
      else
      begin
        CheckNode(node^.LeftChild, level + 1, siblCount);
        CheckNode(node^.RightChild, level + 1, siblCount);
      end;
  end;
begin
  Count := 0;
  CheckNode(Head, 0, Count);
  Result := Count;
end;
```

Локальная процедура CheckNode может использовтся сама по себе для решения задачи. Здесь функция CountSiblings помогает сократить объем основного блока кода и количество глобальных пременных.