```
1   Program bin_tree;
2   {
      Задача.  Построить двоичное дерево, элементами которого являются
               числа.

      Решение. Сергей Митрофанов,
               Центр НИТ "Северная Звезда",
               28.10.03, 11:24-14:21;
               30.11.06, 17:13.
    }
11
12  {
      Теория
      ------
      Двоичное дерево схематично можно определить следующим образом:
      имеется набор вершин, соединенных стрелками-отрезками-ветками.

      Из каждой вершины выходит не более двух стрелок (ветвей),
      направленных влево-вниз или вправо-вниз. Должна существовать
      единственная вершина, из которой не выходит ни одна стрелка -
      эта вершина называется КОРНЕМ дерева.

      В каждую из оставшихся вершин входит в точности одна стрелка.

      Каждое звено (вершина дерева) будет записью из 3 полей.
      1 поле - ключ записи (их последовательность записана в массив A)
      2 поле - адрес на вершину влево-вниз
      3 поле - адрес на вершину вправо-вниз

      Способ построения дерева:
      Если поступает очередная запись с ключом k, то начиная с корня
      дерева в зависимости от сравнения ключа k с ключом в очередной
      вершине, идем влево или вправо от нее, до тех пор, пока не
      найдем
      подходящую вершину, к которой можно присоединить новую вершину с
      ключом k. В зависимости от результата сравнения ключа в этой
      вершине с поступившим ключом k делаем вновь сформированную
      вершину левой или правой для найденной вершины.
    }
40
41  Uses Crt;
42
43  Const
44    A { данный массив-константа }
45    : array [1..12] of integer
46    = (70, 60, 85, 87, 90, 45, 30, 88, 35, 20, 86, 82);
47
48  Type
49    zap = ^element;
50    element = record
```

```
51                    n { число из массива }
52                    : integer;
53                    lev, { адрес левой ветки }
54                    prav { адрес правой ветки }
55                    : zap;
56                  end;
57
58        Var
59          i { номер элемента массива }
60          : integer;
61
62          tek_v, { текущая вершина дерева }
63          lev_v, { левая вершина дерева }
64          prav_v, { правая вершина дерева }
65          root_v, { корень дерева }
66          pred_v, { вершина предыдущая }
67          new_v { новая, только что порожденная вершина с помощью New }
68          : zap;
69
70
71        Procedure sozd_dereva (
72                                k { число-ключ - значение вершины дерева }
73                                : integer
74                              );
75        {
            Алгоритм построения дерева
            MSP, 28.10.03, 12:37
          }

79        var
80          key { ключ - значение вершины дерева }
81          : integer;
82
83        begin
84          {
              если дерево еще не построено, то создаем корень дерева
            }
87          if tek_v = nil
88            then
89              begin
90                New (tek_v);
91                tek_v^.n := k;
92                tek_v^.lev := nil;
93                tek_v^.prav := nil;
94                root_v := tek_v;
95                pred_v := tek_v;

96                Exit;
97              end;
98
99          {
```

```
          добавление следующих вершин дерева
        }
102     {
          поля новой вершины
        }
105     New (new_v);
106     new_v^.n := k;
107     new_v^.lev := nil;
108     new_v^.prav := nil;
109
110     {
          будем двигаться вниз по дереву, влево или вправо и сравнивать
          key с k.
          Цикл закончим тогда, когда придем на вершину, значение которой
          nil.
        }
116     tek_v := root_v;
117     key := root_v^.n; { значение корня дерева }
118     while tek_v <> nil do
119       begin
120         {I. }
121         if (k <= key)
122           then
123             if (tek_v^.lev = nil)
124               then
125                 begin
126                   { записываем адрес новой вершины
                      в текущую левую ветку }
128                   tek_v^.lev := new_v;
129                   WriteLn ('прошла установка ', k, ' налево, после
130                   ', tek_v^.n);

131                   Break;
132                 end
133               else
134                 begin
135                   tek_v := tek_v^.lev;
136                   key := tek_v^.n;

137                   Continue;
138                 end;
139         {II. }
140         if (k > key)
141           then
142             if (tek_v^.prav = nil)
143               then
144                 begin
145                   { записываем адрес новой вершины
                      в текущую правую ветку }
147                   tek_v^.prav := new_v;
148                   WriteLn ('прошла установка ', k, ' направо, после
```

```
149                     ', tek_v^.n);

150                     Break;
151                   end
152                 else
153                   begin
154                     tek_v := tek_v^.prav;
155                     key := tek_v^.n
156                   end;
157
158         end; { end while, конец поиска nil }
159       end; { end sozd_dereva окончание построения дерева }
160
161
162       Procedure del_tree (
163                           p { адрес узла дерева }
164                           : zap
165                         );
166       {
            Уничтожение дерева
          }
169       begin
170         if p <> nil
171           then
172             begin
173               del_tree (p^.lev);
174               del_tree (p^.prav);
175               Dispose (p);
176             end;
177       end;
178
179
180       Begin
181         ClrScr;
182
183         {
              Двоичное дерево еще не построено
            }
186         tek_v := nil;
187
188         WriteLn ('Построим двоичное дерево.');
189         WriteLn;
190         for i := 1 to 12 do
191           begin
192             sozd_dereva (A [i]);
193           end;
194         WriteLn;
195         WriteLn ('Двоичное дерево построено.');
196
197         {
              Уничтожим дерево (отдадим память назад OS)
```

```
            }
200         del_tree (root_v);
201         WriteLn;
202         WriteLn ('Двоичное дерево уничтожено.');
203
204         ReadLn;
205       End.
```

Listing данной задачи опубликован в сети Internet по адресу
http://www.Best-Listing.ru/color-1-task-565.html

Sergey Mitrofanov, 24.08.14, 19:17
E-mail: infostar@mail.ru