```
1       /*
          Задача.  Создание и заполнение двоичного дерева.
                   Сортировка последовательности чисел
                   с помощью двоичного дерева.
                   Удаление дерева

          Решение. Сергей Митрофанов,
                   CNIT "North Star",
                   15-19.05.05
                   28.05.05, 18:46 // сортировка с помощью стека
                   22.09.05, 12:43 // алгоритм написан!
                   24.09.05, 15:14 - удаление дерева
        */

13      # include <iostream.h>
14      # include <conio.h>
15      # include <process.h>

16      // узел дерева
17      struct node
18      {
19        int n; // значение узла
20        struct node * L; // левая связь
21        struct node * R; // правая связь
22      };

23      const N = 10; // длина массива
24      int A [N] = {17, -3, 0, 14, -7, 9, 12, 1, 2, 20};

25      int
26        // сколько занято памяти
27        sv = 0,
28        // значение узла
29        n_node,
30        // номер элмента массива
31        i;

32      struct node
33        // вершина дерева
34        * root,
35        // новый узел
36        * new_node,
37        // текущий узел дерева
38        * tek;


39      // обход дерева по алгоритму Подбельского, с. 440
40      void sort_b_tree (struct node *);
41      void delete_b_tree (struct node *);
```

```
42      int main ()
43      {
44        clrscr ();

45        cout << "Дан массив: ";
46        for (i = 0; i < N; ++ i)
47          cout << A [i] << ' ';
48        cout << endl << endl;

49        tek = new struct node;

50        sv += sizeof (struct node);

51        if (tek == NULL)
52        {
53          cout << "Нет пямяти для создания узла дерева..." << endl;

54          getch ();

55          exit (-1);
56        }

57        tek -> n = A [0];
58        tek -> L = NULL;
59        tek -> R = NULL;

60        root = tek;

61        cout << "Занесем в дерево: " << A [0] << endl;
62        // заполним двоичное дерево элементами массива,
63        // породим новые узлы бинарным образом,
64        // произведем предварительную подготовку к сортировке...
65        for (i = 1; i < N; ++ i)
66        {
67          cout << "Занесем в дерево: " << A [i] << endl;

68          new_node = new struct node;

69          sv += sizeof (struct node);

70          if (new_node == NULL)
71          {
72            cout << "Нет пямяти для создания узла дерева..." << endl;
73            getch ();
74            exit (-1);
75          }

76          new_node -> n = A [i];
77          new_node -> L = NULL;
78          new_node -> R = NULL;
```

```
79          tek = root;
80          n_node = root -> n;
81          while (1 == 1)
82          {
83            // ищем самый нижний левый NULL
84            // и прикрепляем на его место новый узел
85            if (A [i] < n_node)
86            {
87              if (tek -> L == NULL)
88              {
89                tek -> L = new_node;

90                //cout << "создана левая ветвь: " << A [i] << endl;

91                break;
92              }
93              tek = tek -> L;
94              n_node = tek -> n;
95            }

96            // ищем самый правый нижний NULL
97            // и прикрепляем на его место новый узел
98            else
99            {
100             if (tek -> R == NULL)
101             {
102               tek -> R = new_node;

103               //cout << "создана правая ветвь: " << A [i] << endl;

104               break;
105             }
106             tek = tek -> R;
107             n_node = tek -> n;
108           }
109         }
110       } // end for (i = 1; i < N; ++ i)

111       // MSP, 22.09.05
112       // отсортируем дерево
113       sort_b_tree (root);

114       // MSP, 24.09.05, 15:04
115       delete_b_tree (root);

116       getch ();

117       return 0;
118     } // end int main ()
```

```
119     void sort_b_tree (struct node * uzel)
120     {
121       // MSP, 22.09.05, 14:50
122       // Обход дерева, сортировка дерева.
123       // Алгоритм взят из книги:
124       // Подбельский В.В., Фомин С.С. Программирование на языке Си,
125       // с.440

126       // закажем огромный стек под запоминание СОТНИ уровней дерева
127       const int U = 100; // число уровней дерева
128       struct node * stack [U];

129       // очистим стек
130       for (i = 0; i < U; ++ i)
131         stack [i] = NULL;

132       cout << endl << "Ответ: ";

133       // обойдем дерево, отсортируем данные

134       // номер уровня дерева
135       int i = 0;
136       while (1 == 1)
137       {
138         // найдем самый нижний NULL-узел
139         while (uzel != NULL)
140         {
141           // все встречающиеся на нашем пути узлы записываем в стек
142           stack [i] = uzel;
143           ++ i;
144           // постоянно идем только по левым ветвям
145           uzel = uzel -> L;
146         }

147         // левый NULL-узел найден, заберем из массива-стека верхний
148         // элемент
149         // зададим номер верхнего элемента стека
150         -- i;
151         // если весь стек просмотрен, то и все дерево просмотрено,
152         // пора выйти из внешего цикла
153         if (i < 0)
154           break;

155         uzel = stack [i];
156         // приготовим указатель стека к следующему забору элемента


157         // печатаем верхний элемент стека
158         cout << uzel -> n << " ";

159         // посмотрим, а что у текущего узла в правом указателе?
```

```
160         uzel = uzel -> R;
161       } // конец просмотра-обхода дерева

162       cout << endl;

163       cout << "Под дерево занято: " << sv << " bytes" << endl;
164     } // end void sort_b_tree (struct node * uzel)


165     void delete_b_tree (struct node * uzel)
166     {
167       // MSP, 24.09.05, 14:44
168       // Обход дерева, удаление дерева.

169       // Алгоритм обхода взят из книги:
170       // Подбельский В.В., Фомин С.С. Программирование на языке Си,
171       // с.440

172       // закажем огромный стек под запоминание СОТНИ уровней дерева
173       const int U = 100; // число уровней дерева
174       struct node * stack [U];
175       struct node * temp; // адрес узла, который надо удалить

176       // очистим стек
177       for (i = 0; i < U; ++ i)
178         stack [i] = NULL;

179       uzel = root;

180       cout << endl << "Удаляем дерево... " << endl;

181       // обойдем дерево, удалим его вершины

182       // номер уровня дерева
183       int i = 0;
184       while (1 == 1)
185       {
186         // найдем самый нижний NULL-узел
187         while (uzel != NULL)
188         {
189           // все встречающиеся на нашем пути узлы записываем в стек
190           stack [i] = uzel;
191           ++ i;
192           // постоянно идем только по левым ветвям
193           uzel = uzel -> L;
194         }

195         // левый NULL-узел найден, заберем из массива-стека верхний
196         // элемент
197         // зададим номер верхнего элемента стека
```

```
198         -- i;
199         // если весь стек просмотрен, то и все дерево просмотрено,
200         // пора выйти из внешего цикла
201         if (i < 0)
202           break;

203         uzel = stack [i];
204         // приготовим указатель стека к следующему забору элемента

205         // печатаем верхний элемент стека
206         cout << "удаляем вершину с " << uzel -> n << endl;

207         // сохранил адрес узла, который надо удалить
208         temp = uzel;

209         // посмотрим, а что у текущего узла в правом указателе?
210         uzel = uzel -> R;

211         delete temp;
212         sv -= sizeof (struct node);
213       } // конец просмотра-обхода дерева

214       cout << "Под дерево выделено: " << sv << " bytes" << endl;
215     } // end void delete_b_tree (struct node * uzel)
```

Listing данной задачи опубликован в сети Internet по адресу
http://www.Best-Listing.ru/color-3-task-21.html

Sergey Mitrofanov, 31.10.13, 15:17
E-mail: infostar @ mail.ru