

```

1  struct Node
2  {
3      int data;
4      Node *pLeft;
5      Node *pRight;
6  };
7
8  struct Tree
9  {
10     Node *pRoot;
11 };
12
13 Node *initNode(int value)
14 {
15     Node *p = new Node;
16     p->data = value;
17     p->pLeft = NULL;
18     p->pRight = NULL;
19
20     return p;
21 };
22
23 /* Cau 3: Viết hàm khởi tạo cây.
24 */
25 void initTree(Tree &t)
26 {
27     t.pRoot = NULL;
28 };
29
30 void addNode(Tree &t, Node *p)
31 {
32     if (t.pRoot == NULL)
33     {
34         t.pRoot = p;
35     }
36     else
37     {
38         Node *pGoto = t.pRoot;
39         Node *pLoca = NULL;
40
41         while (pGoto != NULL)
42         {
43             pLoca = pGoto;
44             if(p->data == pGoto->data)
45                 return;
46
47             if (p->data < pGoto->data)
48                 pGoto = pGoto->pLeft;
49             else if (p->data > pGoto->data)
50                 pGoto = pGoto->pRight;
51         }
52
53         if (p->data < pLoca->data)
54         {
55             pLoca->pLeft = p;
56         }
57         else if (p->data > pLoca->data)
58         {
59             pLoca->pRight = p;
60         }
61     }
62 };
63

```

```

63
64 void addValue(Tree &t, int value)
65 {
66     Node *p = initNode(value);
67     addNode(t, p);
68 };
69
70 int initDataTreeAuto(Tree &t)
71 {
72     srand(time(0));
73     int n = rand() % (20 - 10 + 1) + 10;
74
75     for (int i = 0; i < n; i++)
76     {
77         int value = rand() % (68 - -38 + 1) + -38;
78         addNode(t, initNode(value));
79     }
80     return n;
81 }
82
83 void initDataTreeArr(Tree &t, int arr[], int n)
84 {
85     for (int i = 0; i < n; i++)
86     {
87         addNode(t, initNode(arr[i]));
88     }
89 };
90
91 void printTree(Tree t)
92 {
93     stack<Node *> s;
94     Node *p = t.pRoot;
95
96     while (p != NULL || s.empty() == false)
97     {
98         // Left
99         while (p != NULL)
100         {
101             s.push(p);
102             p = p->pLeft;
103         }
104
105         // Get the top node and delete it
106         p = s.top();
107         s.pop();
108
109         // Print the node
110         cout << p->data << " ";
111
112         // Move to the right node
113         p = p->pRight;
114     }
115     cout << endl;
116     << endl;
117 }
118
119 bool timGiaTri(Tree t, int value)
120 {
121     Node *p = t.pRoot;
122
123     while (p != NULL)
124     {
125         if (value == p->data)
126             return true;
127
128         if (value < p->data)
129             p = p->pLeft;
130         else if (value > p->data)
131             p = p->pRight;
132     }
133     return false;
134 };

```