```
1 #include <iostream>
2 using namespace std;
4 //
5 //顺序查找
6 //顺序表
7 int Search(int arr[], int n, int key)
8 {
9
       int i;
       for (i = 0; i < n; i++)
10
11
          if (arr[i] == key)
12
              return 1;
13
       return -1;
14 }
15 //链式表
16 LNode* Search(LNode* head, int key)
17 {
18
       LNode* p = head->next;
19
       while (p != NULL)
20
21
           if (p-)next == key)
22
              return p;
23
           p = p- next;
24
25
       return NULL;
26 }
27 //折半查找(适用有序数组)
28 int BSearch(int arr[], int low, int high, int key)
29 {
30
       while (low <= high)</pre>
31
32
           int mid = (1ow + high) / 2;
           if (arr[mid] == key)
33
34
              return mid;
35
           else if (arr[mid] > key)
36
              high = mid - 1;
37
           else
38
              low = mid + 1;
       }
39
40
       return -1;
41 }
42 //分块查找(适用含有一定有序性的数组,分块,利用折半查找确定范围,再利用顺序查找找到元 >
    素)
43 typedef struct
44 {
45
       int maeKey;
46
       int low, high;
47
48 }indexElem;
49 indexElem index[5];
50 int keys[15];
51 //二叉排序树
52 //查序1.
53 BTNode* BSTSearch (BTNode* p, int key)
54 {
55
       while (p != NULL)
```

```
56
57
             if (key == p-)key)
58
                 return p;
59
             else if (key < p->key)
60
                 p = p \rightarrow 1Child;
61
62
                 p = p \rightarrow rChild;
63
64
         return NULL;
65 }
66 //2.
67 BTNode* BSTSearch(BTNode* p, int key)
68 {
69
         if (p == NULL)
70
            return NULL;
 71
         else
 72
 73
             if (p-)key == key)
74
                 return p;
75
             else if (key < p->key)
 76
                 return BSTSearch(p->1Child, key);
77
             else
78
                 return BSTSearch(p->rChild, key);
 79
80 }
81 //插入关键字
82 int BSTInsert (BTNode*& p, int key)
84
         if (p == NULL)
85
86
             p = (BTNode*)malloc(sizeof(BTNode));
87
             p->1Child = p->rChild = NULL;
88
             p->key = key;
89
             return 1;
90
91
         }
92
         else
93
94
             if (key == p-)key)
95
                 return 0;
96
             else if (key < p->key)
97
                 return BSTInsert(p->1Child, key);
98
             else
99
                 return BSTInsert(p->rChild, key);
100
101 }
102 //删除关键字
103 //具体情况具体分析吧
104
105 int main()
106 {
107
108 }
```