

```
1  #include <iostream>
2  using namespace std;
3
4  //
5  //顺序查找
6  //顺序表
7  int Search(int arr[], int n, int key)
8  {
9      int i;
10     for (i = 0; i < n; i++)
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)
31     {
32         int mid = (low + high) / 2;
33         if (arr[mid] == key)
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 } indexElem;
48 indexElem index[5];
49 int keys[15];
50 //二叉排序树
51 //查序1.
52 BTNode* BSTSearch(BTNode* p, int key)
53 {
54     while (p != NULL)
```

```
56     {
57         if (key == p->key)
58             return p;
59         else if (key < p->key)
60             p = p->lChild;
61         else
62             p = p->rChild;
63     }
64     return NULL;
65 }
66 //2.
67 BTreeNode* BSTSearch(BTreeNode* 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->lChild, key);
77         else
78             return BSTSearch(p->rChild, key);
79     }
80 }
81 //插入关键字
82 int BSTInsert(BTreeNode*& p, int key)
83 {
84     if (p == NULL)
85     {
86         p = (BTreeNode*)malloc(sizeof(BTreeNode));
87         p->lChild = p->rChild = NULL;
88         p->key = key;
89         return 1;
90     }
91     else
92     {
93         if (key == p->key)
94             return 0;
95         else if (key < p->key)
96             return BSTInsert(p->lChild, key);
97         else
98             return BSTInsert(p->rChild, key);
99     }
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 }
```