

北京交通大学

《数据结构（A）》第9章

“查找”基本作业与设计作业

专 业： 计算机科学与技术

班 级：

学生姓名：

学 号：

北京交通大学计算机与信息技术学院

2021 年 12 月 18 日

《数据结构 (A)》第 9 章基本与设计作业^①

提醒同学：本章基本作业与设计型作业合并在一起，即没有另外专门的设计作业题目。

1 作业题目

9.1 已知如下所示长度为 12 的表

(Jan, Feb, Mar, Apr, May, June, July, Aug, Sep, Oct, Nov, Dec)

(1) 试按表中元素的顺序依次插入一棵初始为空的二叉排序树，画出插入完成后的二叉排序树，并求其在等概率的情况下查找成功的平均查找长度。

(2) 若对表中元素先进行排序构成有序表，求在等概率情况下对此有序表进行折半查找时查找成功的平均查找长度。

(3) 按表中元素顺序构造一棵平衡二叉排序树，并求其在等概率的情况下查找成功的平均查找长度。

9.2 试从空树开始，画出按以下次序向 2-3 树即 3 阶 B-树中插入关键码的建树过程：20、30、50、52、60、68、70，如果此后删除 50 和 68，画出每一步执行后 2-3 树的状态。

9.3 选取哈希函数 $H(k) = (3k) \text{ MOD } 11$ 。用开放定址法处理冲突， $di = i ((7k) \text{ MOD } 10 + 1) (i=1, 2, 3\cdots)$ 。试在 0—10 的散列地址空间中对关键字序列 (22、41、53、46、30、13、01、67) 构造哈希表，并求等概率情况下查找成功时的平均查找长度。

^① 这是《数据结构 (A)》第 9 章的基本与设计作业，第 14 周周一发布 (2021 年 12 月 13 日星期一)，学生提交的截止日期是 2021 年 12 月 26 日。

9.4 试为下列关键字建立一个装载因子不小于 0.75 的哈希表，并计算你所构造的哈希表的平均查找长度。（ZHAO、QIAN、SUN、LI、ZHOU、WU、ZHANG、WANG、CHANG、CHAO、YANG、JIN）

9.5 在地址空间为 0—16 的散列区中，对以下关键字序列构造两个哈希表：

(Jan, Feb, Mar, Apr, May, June, July, Aug, Sep, Oct, Nov, Dec)

(1) 用线性探测开放定址法处理冲突

(2) 用链地址法处理

并分别求这两个哈希表在等概率情况下查找成功和不成功时的平均查找长度。设哈希函数为 $H(x) = i/2$ ，其中 i 为关键字中第一个字母在字母表中的序号。

9.6 设计一个读入一串整数构成一颗二叉排序树的程序，从二叉排序树中删除一个结点，使该二叉树仍保持二叉排序树的特性。

9.7 设定哈希函数 $H(\text{key}) = \text{key} \bmod 11$ （表长=11），输入一组关键字序列，根据线性探测再散列解决冲突的方法建立哈希表的存储结构，显示哈希表，任意输入关键字，判断是否在哈希表中。

2 作业题目解答

【9.1 题解答】

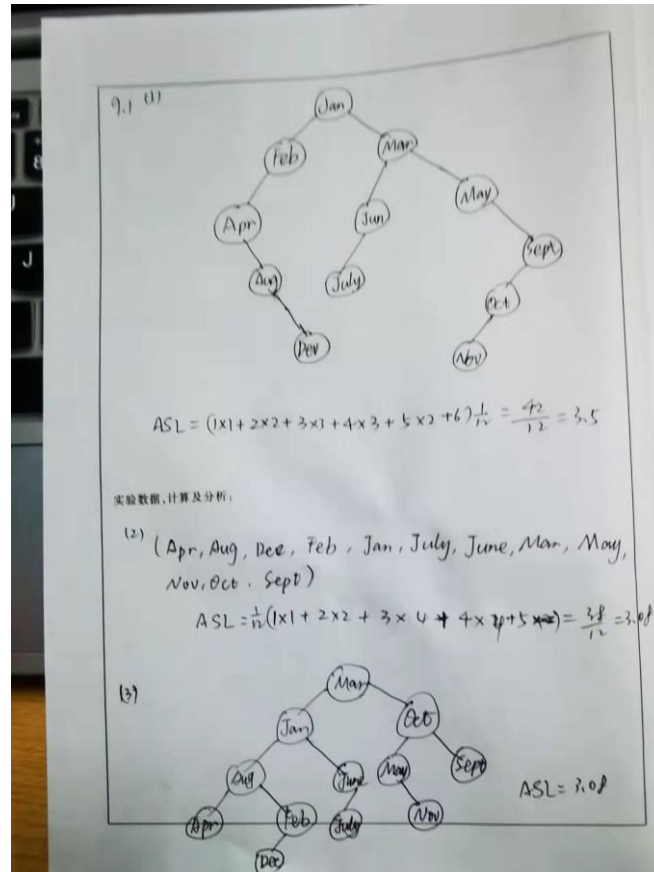


图 9.1-1

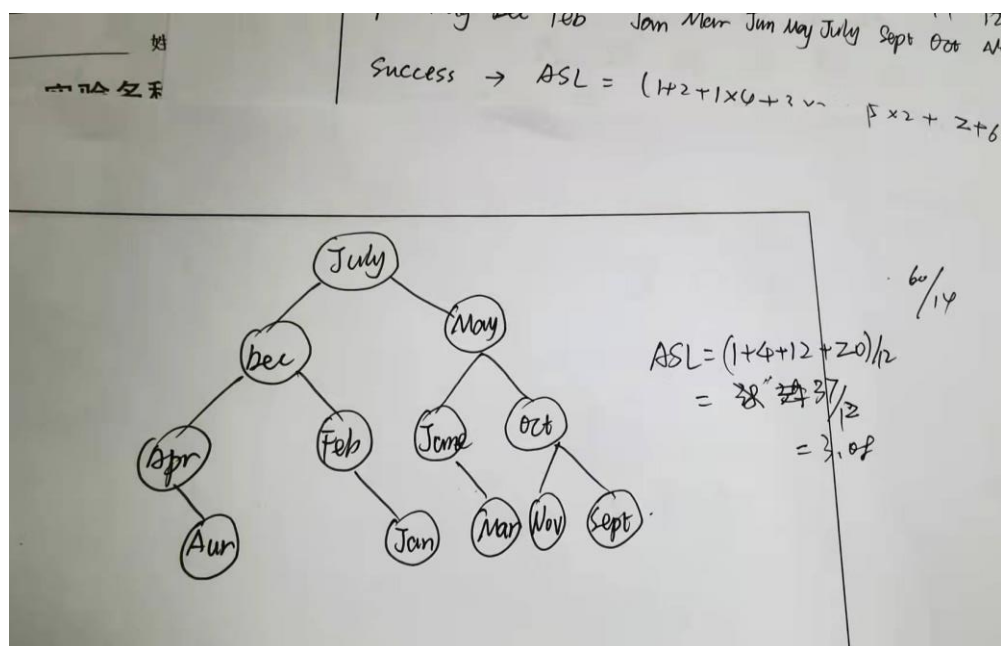


图 9.1-2

【9.2 题解答】

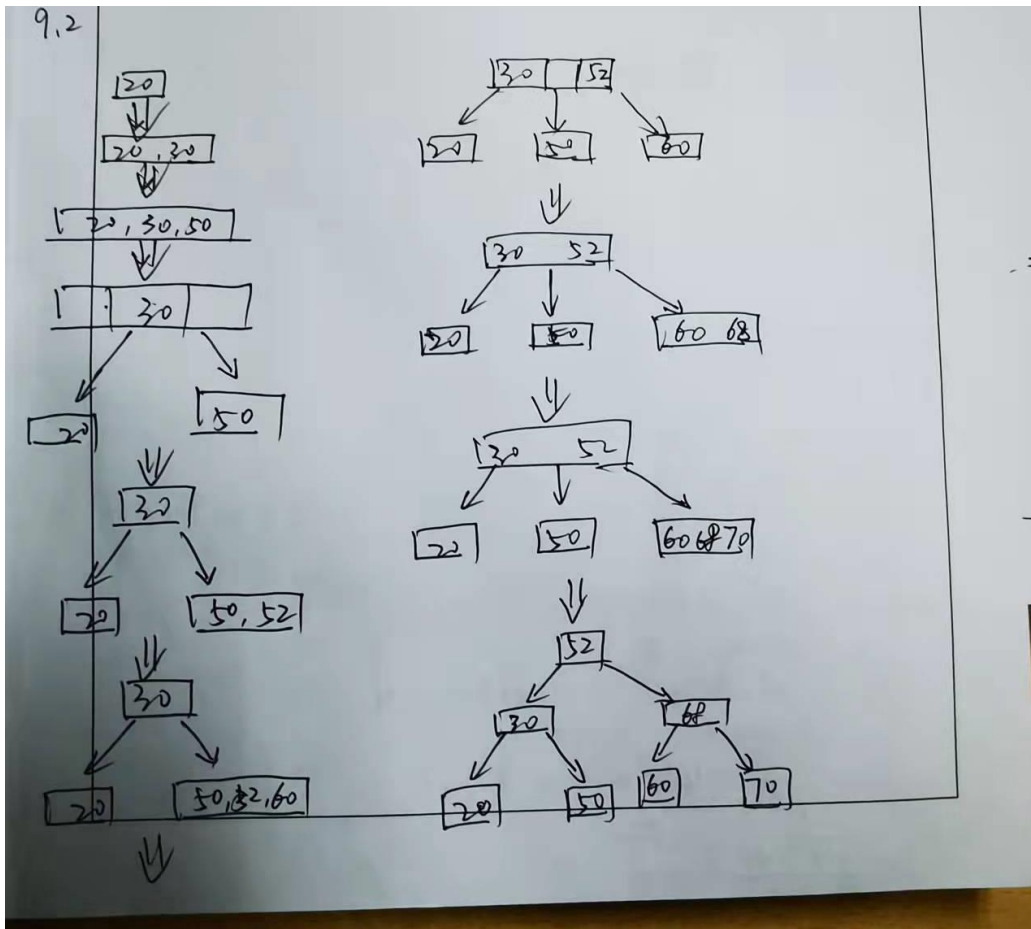


图 9.2-1

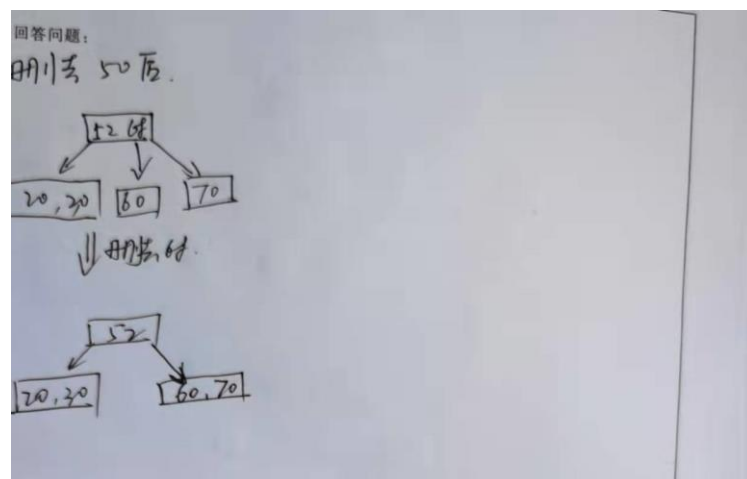


图 9.2-2

【9.3 题解答】

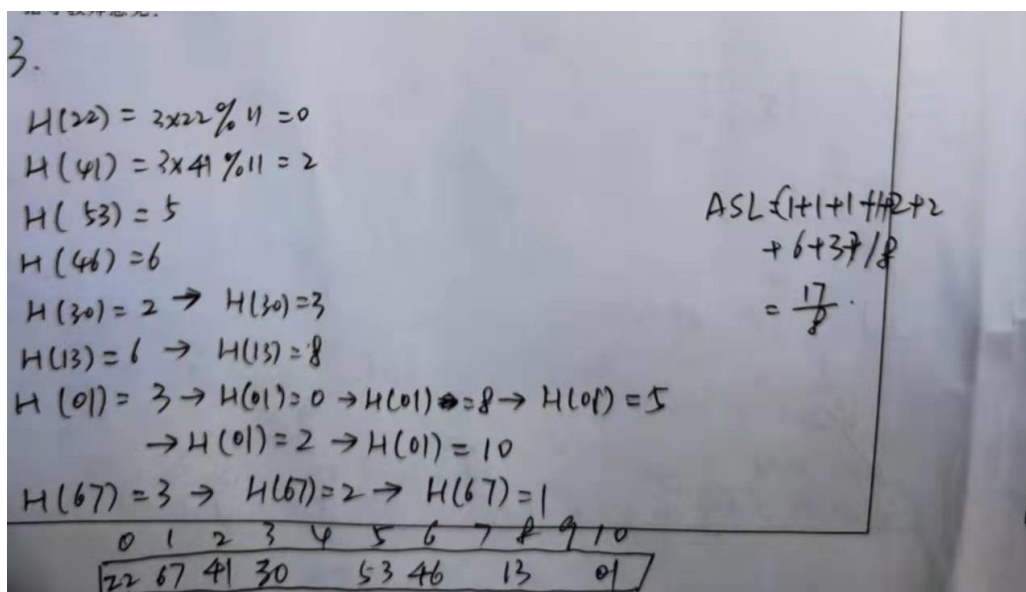


图 9.3

【9.4 题解答】

思路:

$$12/0.75 = 16$$

表长取 16.

先把字符串转化为 int 型量化, 再正常制作哈希表。

转化公式为

(伪代码)

```
For (int I = 0; i < 16; ++i)
{
    num = 10 * num + (str[i][0] - 'A')
}
```

以模十六为哈希函数, 冲突则加一。

运算量过大, 自动化实现。

代码:

```
01:char str[12][6] = {"ZHAO", "QIAN", "SUN", "LI", "ZHOU", "WU", "ZHANG",
02:"WANG",
03:          "CHANG", "CHAO", "YANG", "JIN"};
04:typedef struct hash_table{
05:    int situation;
06:    char data[10];
07:};
08:int insert_hash(hash_table table[], int key, int order){
09:    int index = key % 16;
10:    int i;
11:    if(table[index].situation ==0 ) strcpy(table[index].data,str[order]);
12:    for(i= 0; table[index].situation!=0 && i <16; i++)
13:        index = (index + 1) % 16;
14:    if(i ==16) return 0;
15:    table[index].situation =1;
16:    strcpy(table[index].data, str[order]);
17:    return 1;
18:}
19:
20:void show_hash(hash_table table[]){
21:    for(int i =0; i<16 ; ++i){
22:        printf("%5d", i);
23:    }
24:    printf("\n");
25:    for(int i =0 ; i<16; ++i){
26:        if(table[i].situation ==1)
27:            printf("%5s", table[i].data);
28:        else
29:            printf("   ");
30:    }
31:}
32:hash_table table[16];
33:    for (int i = 0; i < 16; ++i) {
34:        table[i].situation = 0;
35:    }
36:int num[12];
37:int s= 0;
38:for(int i= 0; i<12; ++i){
39:    num[i] = 10 * s + str[i][0]-'A';
40:    s = num[i];
41:}
42:for(int i = 0; i<16; ++i){
43:    insert_hash(table, num[i], i);
```



```
44:}
45:  show_hash(table);
```

调试正常

```
"D:\BJTU\subject\sophomore\data structure\chapter_9\cmake-build-debug\chapter_9.exe"
  0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
WANGZHANG          YANG          SUN   LI CHAO ZHAO QIAN          WU          ZHOU
进程已结束，退出代码为 0
```

图 9.4

【9.5 题解答】

调试正常

9.5.

主要仪器名称: $H(x) = \lfloor i/2 \rfloor$ 冲突加1

$\Rightarrow H(\text{Jan}) = 5$
 $H(\text{Feb}) = 3$
 $H(\text{Mar}) = 6$
 $H(\text{Apr}) = 0$
 $H(\text{June}) = 5 \rightarrow H(\text{June}) = 6 \rightarrow H(\text{June}) = 7$
 $H(\text{May}) = 6 \rightarrow H(\text{May}) = 7 \rightarrow 8$
 $H(\text{July}) = 5 \rightarrow \dots \rightarrow 9$

实验原理, 操作步骤:

$H(\text{Aug}) = 0 \rightarrow 1$
 $H(\text{Oct}) = 7 \dots \rightarrow 11$
 $H(\text{Nov}) \dots 12$
 $H(\text{Dec}) = 2$

图 9.5-1

0 1 2 3 4 5 6 7 8 9 10 11 12
 Apr Aug Dec Feb Jan Mar Jun May July Sept Oct Nov

Success $\rightarrow ASL = (1 \times 2 + 1 \times 4 + 3 \times 2 + 5 \times 2 + 2 \times 6) / 12$
 $= 31/12$

失败: 空为止. $[0, 13] \in [i/2]$
 $(5 + 4 + 3 + 2 + 1 + 9 + \dots + 1) / 14 = 60/14$

(2).

取 $h(x)_1$

实验数据, 计算及分析: \rightarrow Oct \rightarrow Nov

AST. Success: $(\cancel{2+1+1} + 3+2 + \cancel{2+1}) / 12 = 18/12$

Failure:
 与空比较. 不算有.
 $(\cancel{2+1+1} + 3+2 + \cancel{2+1}) / 14 = 12/14$

图 9.5-2

【9.6 题解答】

思路:

建立 **btree** 结构, 完成搜索和插入函数。删除函数分三种情况讨论, 只用左或右子树、满树和叶子节点。

代码:

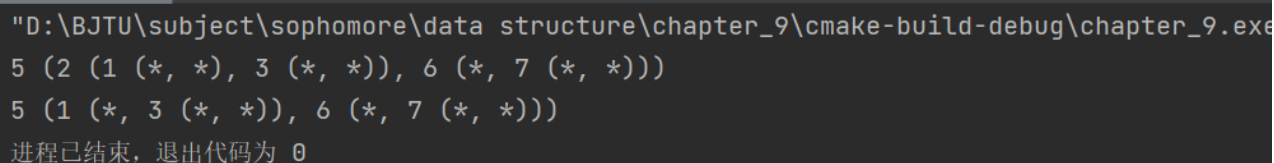
```
46:#include <stdio.h>
47:#include <stdlib.h>
48:
49:
50:typedef struct Btree{
51:    int data;
52:    Btree * lchild;
53:    Btree * rchild;
54:};
55:
56:
57:int search_btree(Btree *&T, int target, Btree *&last_traversal, Btree
    *&before_last)
58:{
59:    if(!T) return 0;
60:    if(target == T->data){
61:        before_last = last_traversal;
62:        last_traversal = T;
63:        return 1;
64:    }
65:    else if(target > T->data)
66:    {
67:        before_last = last_traversal;
68:        last_traversal = T;
69:        search_btree(T->rchild, target, last_traversal, before_last);
70:    }
71:    else{
72:        before_last = last_traversal;
73:        last_traversal = T;
74:        search_btree(T->lchild, target, last_traversal, before_last);
75:    }
76:}
77:
78:int insert_btree(Btree * &T, int target){
79:    Btree* last =(Btree*) malloc(sizeof (Btree));
80:    Btree* before =(Btree*) malloc(sizeof (Btree));
```

```
81:
82:   if(!search_btree(T, target, last, before)){
83:       if(!T){
84:           T = (Btree*) malloc(sizeof (Btree));
85:           T->data = target;
86:           T->rchild = T->lchild = NULL;
87:           return 1;
88:       }
89:
90:       Btree* tem = (Btree*) malloc(sizeof (Btree));
91:       tem->data = target;
92:       tem->rchild = tem->lchild = NULL;
93:
94:       if(target > last->data)
95:           last->rchild = tem;
96:       else
97:           last->lchild = tem;
98:       return 1;
99:   }
100:  else
101:      return 0;
102:}
103:void show_btree(Btree* T)
104:{
105:   if(!T) printf("");
106:   else {
107:       printf("%d ",T->data);
108:       printf("(");
109:       show_btree(T->lchild);
110:       printf(", ");
111:       show_btree((T->rchild));
112:       printf(")");
113:   }
114:}
115:
116:int delete_btree(Btree *T, int target)
117:{
118:   Btree* last =(Btree*) malloc(sizeof (Btree));
119:   Btree* before =(Btree*) malloc(sizeof (Btree));
120:   last->lchild = last->rchild = before->lchild = before->rchild = NULL;
121:
122:   if(!search_btree(T, target, last, before)) return 0;
123:   else{
```

```
124:     Btree * tem;
125:     if( !last->lchild && !last->rchild){
126:         if(before->data > last->data)
127:             before->lchild = NULL;
128:         else
129:             before->rchild = NULL;
130:         free(last);
131:     }
132:     else if(last->lchild == NULL)
133:     {
134:         tem = last->rchild;
135:         *last = *last->rchild;
136:         free(tem);
137:     }
138:     else if(last->rchild == NULL)
139:     {
140:         tem = last->lchild;
141:         *last = *last->lchild;
142:         free(tem);
143:     }
144:     else{
145:         Btree *q=last;
146:         tem = last->lchild;
147:         while(tem->rchild){
148:             q = tem;
149:             tem = tem->rchild;
150:         }
151:         last->data = tem->data;
152:         if(q == last){
153:             q->lchild = tem->lchild;
154:         }
155:         else{
156:             q->rchild = tem->lchild;
157:         }
158:         free(tem);
159:     }
160: }
161:}
162:
163:
164:
165:int main() {
166:    int num_array[10] = {5, 2, 6, 1, 3, 7,};
```

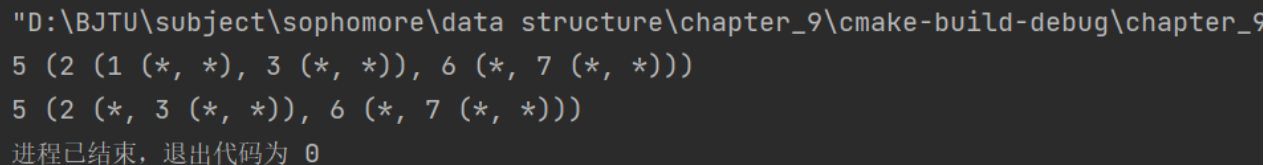
```
167: Btree *T = NULL;
168:
169: for (int i = 0; i < 6; ++i) {
170:     insert_btree(T, num_array[i]);
171: }
172: show_btree(T);
173: printf("\n");
174: delete_btree(T, 1);
175: show_btree(T);
}
```

调试正常:



```
"D:\BJTU\subject\sophomore\data structure\chapter_9\cmake-build-debug\chapter_9.exe
5 (2 (1 (*, *), 3 (*, *)), 6 (*, 7 (*, *)))
5 (1 (*, 3 (*, *)), 6 (*, 7 (*, *)))
进程已结束, 退出代码为 0
```

图 9.6-1



```
"D:\BJTU\subject\sophomore\data structure\chapter_9\cmake-build-debug\chapter_9
5 (2 (1 (*, *), 3 (*, *)), 6 (*, 7 (*, *)))
5 (2 (*, 3 (*, *)), 6 (*, 7 (*, *)))
进程已结束, 退出代码为 0
```

图 9.6-2

【9.7 题解答】

思路:

坦白的说, 可能是本人学艺不精, “线性探测再散列” 似乎上课时没有提到过, 网上也众说纷纭, 有的说就是开放地址法, 有的说是再哈希法。这里采用开放地址法, 冲突处理时, **index** 一次加 1。

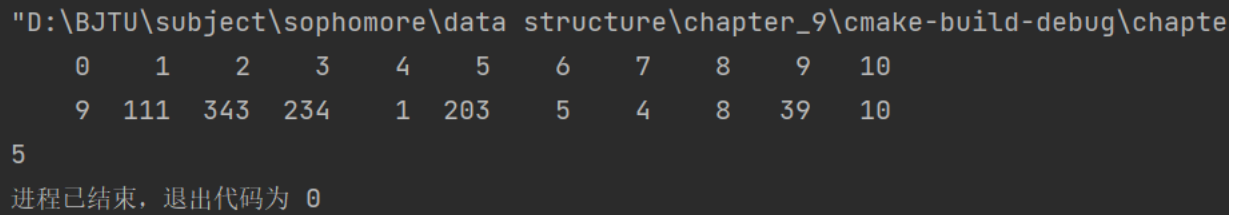
代码:

```
01:typedef struct hash_table{
02:    int situation;
03:    int data;
04:};
```

```
05:int insert_hash(hash_table table[], int key){
06:    int index = key % 11;
07:    int i;
08:    if(table[index].situation ==0 ) table[index].data = key;
09:    for(i= 0; table[index].situation!=0 && i <11; i++)
10:        index = (index + 1) % 11;
11:    if(i ==11) return 0;
12:    table[index].situation =1;
13:    table[index].data = key;
14:    return 1;
15:}
16:
17:void show_hash(hash_table table[]){
18:    for(int i =0; i<11 ; ++i){
19:        printf("%5d", i);
20:    }
21:    printf("\n");
22:    for(int i =0 ; i<11; ++i){
23:        if(table[i].situation ==1)
24:            printf("%5d", table[i].data);
25:        else
26:            printf("  ");
27:    }
28:}
29:int search_hash(hash_table table[], int key){
30:    int index = key % 11;
31://    if(table[index].situation == 0) return -1;
32://    if(table[index].data == key) return index;
33:    while(table[index].situation == 1 && index<11){
34:        if(table[index].data == key) return index;
35:        index++;
36:    }
37:    return -1;
38:}
39: hash_table table[11];
40: for (int i = 0; i < 11; ++i) {
41:     table[i].situation =0;
42: }
43: int key[11] = {111, 203, 234, 343, 1 ,5, 8, 4, 39, 10, 9};
44: for (int i = 0; i < 11; ++i) {
45:     insert_hash(table, key[i]);
46: }
47: show_hash(table);
```

```
48:    printf("\n%d", search_hash(table, 203));
```

调试正常:



```
"D:\BJTU\subject\sophomore\data structure\chapter_9\cmake-build-debug\chapt
  0    1    2    3    4    5    6    7    8    9   10
  9  111  343  234    1  203    5    4    8   39   10
5
进程已结束，退出代码为 0
```

图 9.7