# 实验6 查找

**实验日期:** 2020年6月23日 周吉瑞 20190521340**报告日期:** 2020年6月23日

### 一、实验目的

1、掌握查找表、动态查找表、静态查找表和平均查找长度的概念。

2、掌握线性表中顺序查找和折半查找的方法。

3、掌握二叉排序树的建立及查找方法。

### 二、实验预习

复习以下概念

1、顺序查找:从表的一端开始,依次将记录的关键字和给定的值进行比较,若某个记录的关键字和给定值相等,则查找成功;反之,若扫描整个表后,仍未找到关键字和给定值相等的记录,则查找失败.

2、折半查找:从表的中间记录开始,如果给定值和中间记录的关键字相等,则查找成功;如果给定值大于或者小于中间记录的关键字,则在表中大于或者小于中间记录的那一半中查找,这样重复操作,直到查找成功,或者在某一步中查找区间为空,则代表查找失败.

1. 二叉排序树：二叉排序树可以看成是一个有序表,所以在二叉排序树上进行查找和折半查找类似,1.若二叉树为空,则查找是失败,返回空指针.

2.若二叉树非空:当KEY=关键字,则查找成功

当KEY<关键字,则递归查找左子树

当KEY>关键字,则递归查找右子树

### 三、实验内容和要求

1. 静态查找表

依据顺序查找算法和折半查找算法的特点，设计出完整的C源程序，分别对不同的查找表使用这两种查找方法进行测试。并完成下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 查找表 | 关键字 | 顺序查找 | | 折半查找 | |
|  |  | 比较次数 | 查找结果 | 比较次数 | 查找结果 |
| { 8，15，19，26，33 ，41，47，52，64，90 } | 41 | 5 | 6 | 3 | 6 |
| { 8，15，19，26，33 ，41，47，52，64，90 } | 28 | 11 | 0 | 4 | 0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 算法分析： | |  | |  | |

* 顺序查找算法实现代码

#include<stdio.h>

#define MAXSIZE 100

#define OK 1;

int count1;

typedef struct{

int key;//关键字域

}ElemType;

typedef struct{

ElemType \*R;

int length;

}SSTable;

int InitList\_SSTable(SSTable &L)

{

L.R=new ElemType[MAXSIZE];

if (!L.R)

{ printf("初始化错误");

return 0;

}

L.length=0;

return OK;

}

**int Creat\_SSTable(SSTable &L)**

{

int i = 1;

int j;

printf("创建顺序表, 以'0'为结束标志.\n");

do {

scanf("%d", &j);

if (j == 0) {

break;

}

ST.R[i++].key = j;

ST.length++;

} while (1);

printf("\n");

return OK;

}

int Search\_Seq(SSTable ST, int key) //设置监视哨的顺序查找

{

/\*在顺序表ST中顺序查找其关键字等于key的数据元素。若找到，则函数值为

该元素在表中的位置，否则为0

并统计出比较的次数

\*/

int i;

count\_1 = 1;

ST.R[0].key = key;

for (i = ST.length; ST.R[i].key != key; --i) {

count\_1++;

}

return i;

}// Search\_Seq

int main( )

{

SSTable ST;

InitList\_SSTable(ST);

Create\_SSTable(ST);

Output\_table(ST);

int testkey;

int resule;

do {

printf("请输入待查关键字, 输入'0'退出查找.\n");

scanf("%d", &testkey);

printf("\n");

resule = Search\_Seq(ST, testkey, count\_1);

Shou\_End(resule, testkey);

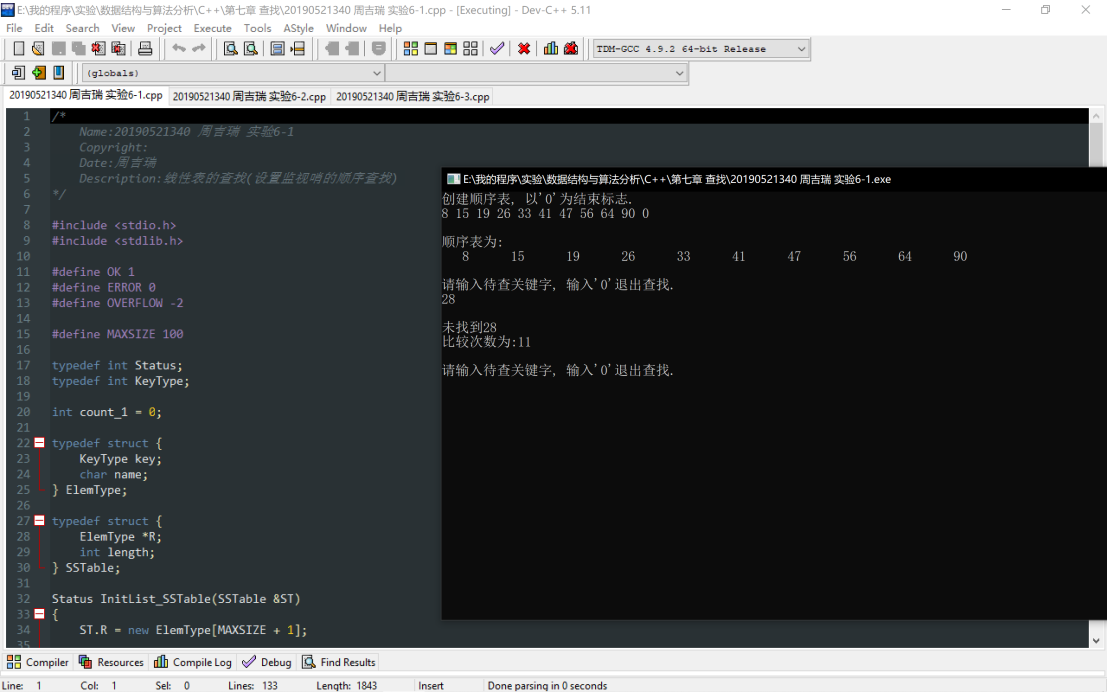
} while (testkey != 0);

return 0;

return 0;

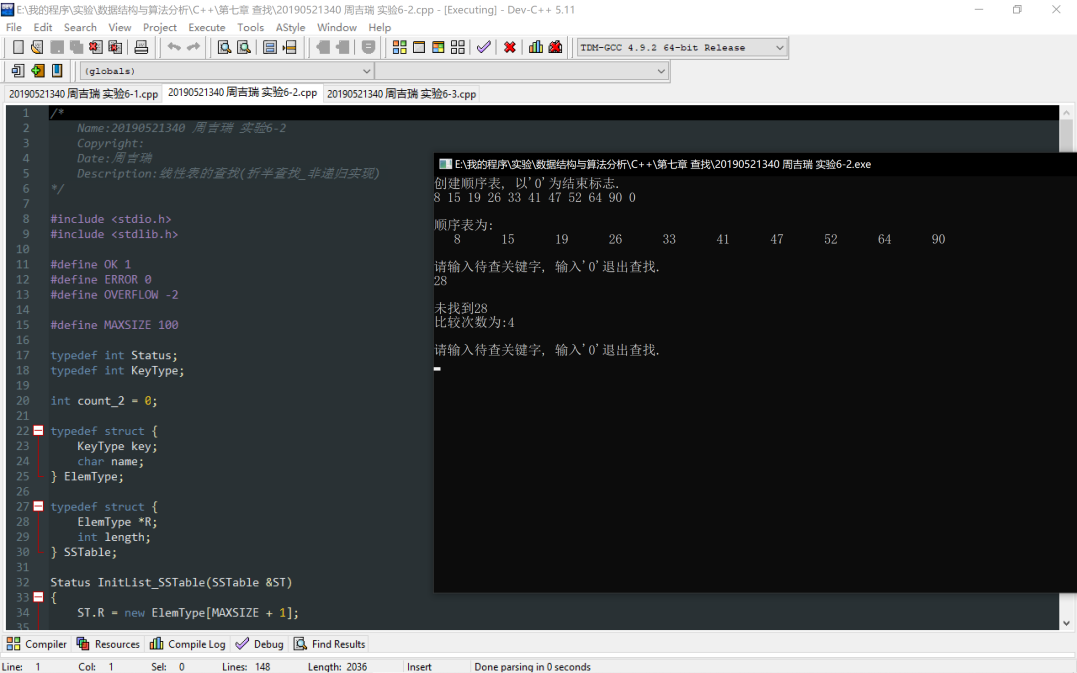
}

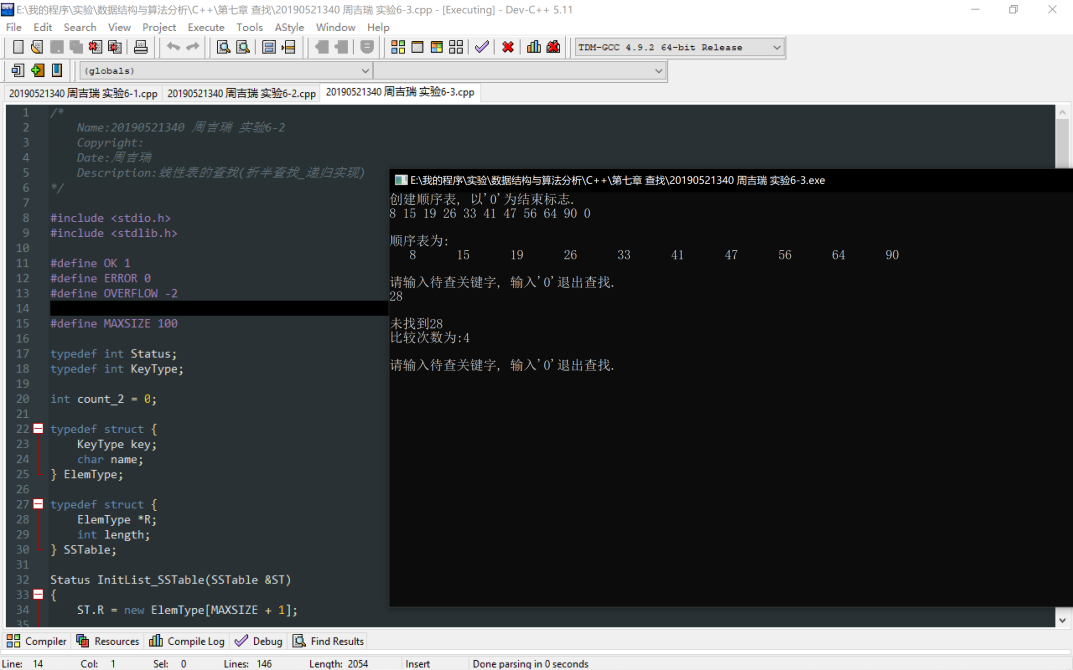
实验结果:



* 折半查找算法实现代码（与顺序查找共用的模块可以省略）

实验结果:





2. （选做）动态查找表（二叉排序树查找）：

试设计建立二叉排序树，并在其中查找元素的算法。

代码:/\*

Name: 20190521340 周吉瑞 实验6-4

Copyright:

Author: 周吉瑞

Date: 26/06/20 08:50

Description: 二叉树表的查找

\*/

#include <stdio.h>

#include <stdlib.h>

#define OK 1;

#define ERROR 0;

#define OVERFLOW -2;

typedef int Status;

typedef int KeyType;

typedef struct {

KeyType key;

char name;

} ElemType;

typedef struct BSTNode {

ElemType data;

struct BSTNode \*lchild, \*rchild;

} BSTNode, \*BSTree;

void InsertBST(BSTree &T, ElemType e);

void CreatBST(BSTree &T);

void InOrder(BSTree T);

BSTree SearchBST(BSTree T, KeyType key);

void Shou\_End(BSTNode \*p);

int main(void)

{

BSTree T;

CreatBST(T);

printf("printf BT:\n");

InOrder(T);

KeyType key;

BSTNode \*p = NULL;

do {

printf("\nSearchBST\n");

printf("input '0' is SearchBST\_end\n");

printf("input key:");

scanf("%d", &key);

getchar();

p = SearchBST(T, key);

Shou\_End(p);

} while (key != 0);

return 0;

}

void InsertBST(BSTree &T, ElemType e)

{

BSTNode \*S;

if (!T) {

S = new BSTNode;

S->data = e;

S->lchild = NULL;

S->rchild = NULL;

T = S;

}

else if (e.key < T->data.key) {

InsertBST(T->lchild, e);

}

else if (e.key > T->data.key) {

InsertBST(T->rchild, e);

}

return ;

} //InsertBST

void CreatBST(BSTree &T)

{

ElemType e;

T = NULL;

printf("CreatBST '0' is end.\n");

scanf("%d", &e.key);

while (e.key != 0) {

InsertBST(T, e);

scanf("%d", &e.key);

} //while

return ;

} //CreatBST

void InOrder(BSTree T)

{

if (T != NULL) {

InOrder(T->lchild);

printf("%d\t", T->data.key);

InOrder(T->rchild);

}

} //InOrder

BSTree SearchBST(BSTree T, KeyType key)

{

if ((!T) || key == T->data.key) {

return T;

}

else if (key < T->data.key) {

return SearchBST(T->lchild, key);

}

else {

return SearchBST(T->rchild, key);

}

} //SearchBST

void Shou\_End(BSTNode \*p)

{

if (p == NULL) {

printf("NO!\n");

}

else {

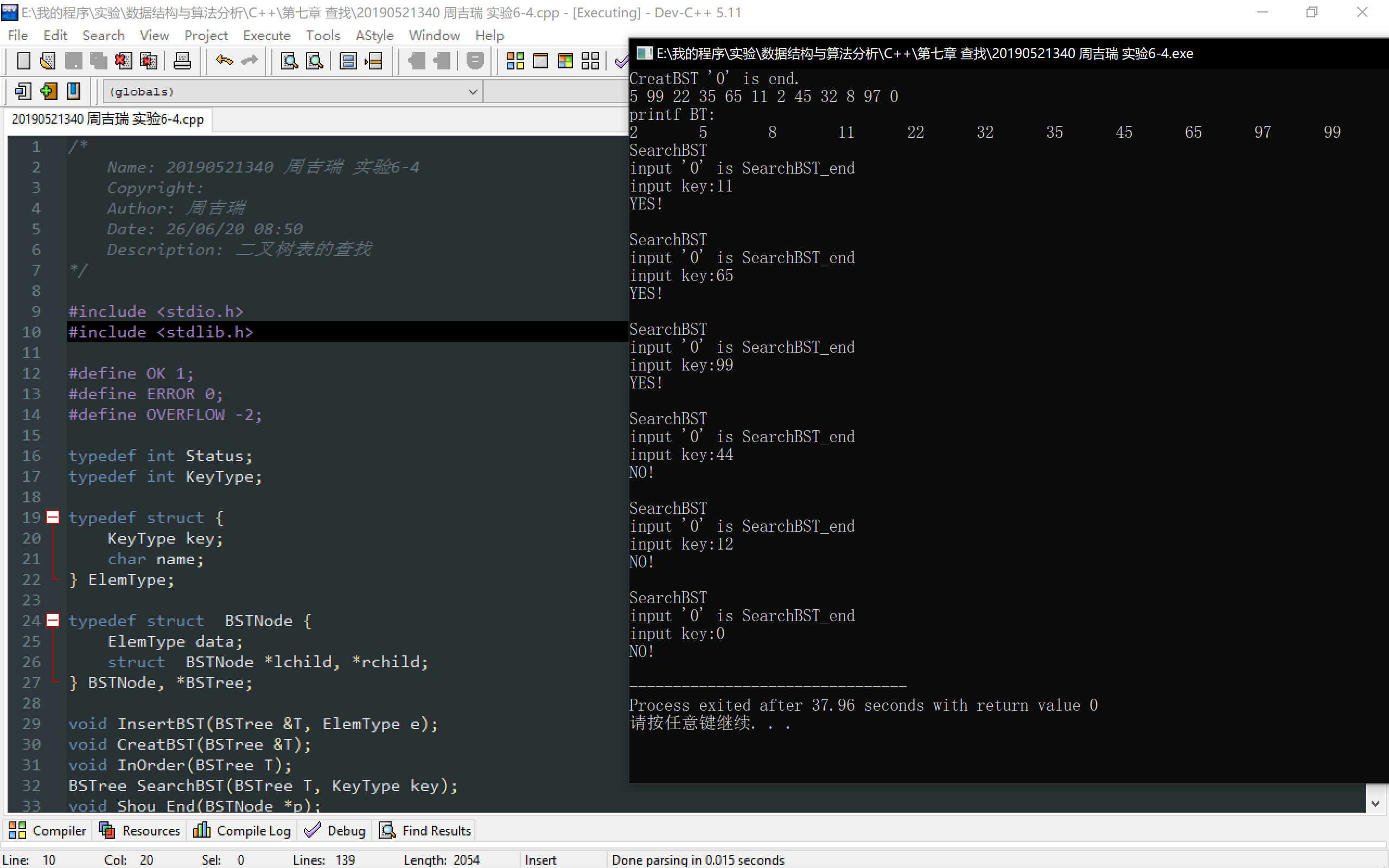
printf("YES!\n");

}

return ;

} //Shou\_End

运行结果:



### 实验小结

1. 在利用折半查找的递归算法时, 计数器发生错误, 最终才发现原因在于:在main函数的do while 循环中 count\_2 多次重复使用, 并且多次使用是并没有对 count\_2清零, 所以应该在main的循环体中加上:count\_2 = 0;
2. 还是那个老问题, 小错误多.