中南大学

数据结构试验报告

题 目 实验六

学生姓名 王云鹏

学 号 8213180228

指导老师 郑 瑾

学 院 计算机学院

专业班级 物联网1802

完成时间 2020.6

指导老师评定 签名

实验六

1. 需求分析

2．二分查找算法（设计性实验）

问题描述

从键盘读入一串整数和一个待查键，查找在该整数串中是否有这个待查键。如果有，就输出它在

整数串中的位置；如果没有，输出−1。

基本要求

掌握二分查找算法。

测试数据

由读者依据软件工程的测试技术自己确定。注意测试边界数据，如单个结点。

实现提示

利用二分查找算法查找实现。

4．简单个人电话号码查询系统（综合性实验）

问题描述

人们在日常生活中经常要查找某个人或某个单位的电话号码，本实验将实现一个简单的个人电话

号码查询系统，根据用户输入的信息（如姓名等）进行快速查询。

基本要求

(1) 在外存上，用文件保存电话号码信息。

(2) 在内存中，设计数据结构存储电话号码信息。

(3) 提供查询功能：根据姓名实现快速查询。

(4) 提供其他维护功能，如插入、删除、修改等。

测试数据

由读者依据软件工程的测试技术自己确定。注意测试边界数据，如单个结点。

实现提示

由于要管理的电话号码信息较多，而且要在程序运行结束后仍然保存电话号码信息，所以电话号码

信息采用文件的形式存放到外存中。在系统运行时，要将电话号码信息从文件调入内存来进行查找等操

作。为了接收文件中的内容，要有一个数据结构与之对应，可以设计如下结构类型的数组来接收数据。

const int max＝10;

struct TeleNumber ｛

String name;// 姓名

String phoneNumber,// 固定电话号码

String mobileNumber,// 移动电话号码

String email;// 电子邮箱

｝Tele[max];

为了实现对电话号码的快速查询，可以将上述结构数组排序，以便应用二分查找，但是，在数组

中实现插入和删除操作的代价较高。如果记录需频繁进行插入或删除操作，可以考虑采用二叉排序树

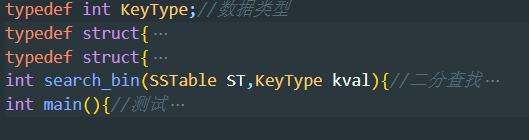
组织电话号码信息，这样查找和维护都能获得较高的时间性能。更复杂地，需考虑该二叉排序树是否

平衡，如何使之达到平衡。

1. 概要设计

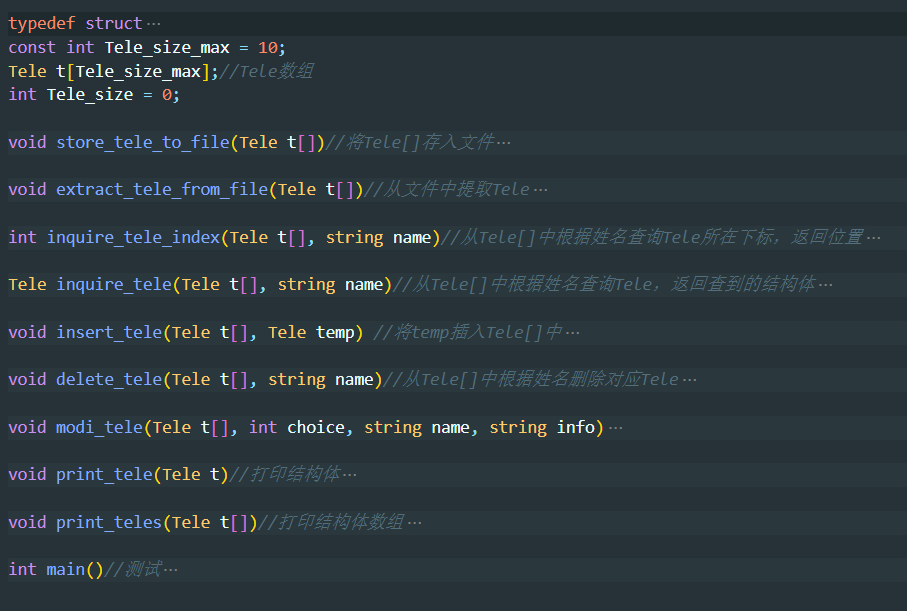
* 设计性实验

函数：



* 综合性实验

函数：



1. 详细设计

* 设计性实验

|  |
| --- |
| *#include*<iostream>  using namespace std;  typedef int KeyType;*//数据类型*  typedef struct{      KeyType key;  }ElemType,TElemType;*//数组中数据类型*  typedef struct{      ElemType \*elem;*//数组*      int length;*//数组中元素个数*  }SSTable;*//存放二分查找的数据类型*  int search\_bin(SSTable ST,KeyType kval){*//二分查找*      int low=0;      int high=ST.length;  *while*(low<=high){*//不断二分*          int mid=(low+high)/2;  *if*(kval==ST.elem[mid].key){*//找到返回位置*  *return* mid;          }*else* *if*(kval<ST.elem[mid].key){*//没找到缩小范围继续寻找*              high=mid-1;          }*else*{              low=mid+1;          }      }  *return* -1;  }  int main(){*//测试*      SSTable ST;      int length;      int kval;      cout<<"输入将要输入的数字个数"<<endl;*//输入数据，初始化要查找的数组*      cin>>length;      ST.length=length;      ST.elem=(ElemType\*)malloc(length\*sizeof(ElemType));      cout<<"输入数字序列"<<endl;  *for*(int i=0;i<length;i++){          cin>>ST.elem[i].key;      }      cout<<"输入要查找的数字"<<endl;      cin>>kval;  *if*(search\_bin(ST,kval)!=-1){          cout<<"要查找的数字位于第"<<search\_bin(ST,kval)+1<<"个"<<endl;      }*else*{          cout<<"要查找的数字不存在"<<endl;      }  *return* 0;  } |

|  |
| --- |
| *#include* <iostream>  *#include* <fstream>  *#include* <cstring>  using namespace std;  typedef struct  {      string name;      string phoneNumber;      string mobileNumber;      string email;  } Tele;*//存放信息的结构体*  const int Tele\_size\_max = 10;  Tele t[Tele\_size\_max];*//Tele数组*  int Tele\_size = 0;  void store\_tele\_to\_file(Tele t[])*//将Tele[]存入文件*  {      ofstream outfile;      outfile.open("./Tele\_info.txt", ios::out);*//打开文件*  *for* (int i = 0; i < Tele\_size; i++)*//向文件写入数据*      {          outfile << t[i].name << " " << t[i].phoneNumber << " " << t[i].mobileNumber << " " << t[i].email << endl;      }      outfile.close();*//关闭文件*      cout<<"写入成功"<<endl;  }  void extract\_tele\_from\_file(Tele t[])*//从文件中提取Tele*  {      ifstream infile;      infile.open("./Tele\_info.txt", ios::in);*//打开文件*  *while*(!infile.eof()){*//从文件中读出数据*          infile >> t[Tele\_size].name >> t[Tele\_size].phoneNumber >> t[Tele\_size].mobileNumber >> t[Tele\_size].email;          Tele\_size++;      }      infile.close();*//关闭文件*      cout<<"导入成功"<<endl;  }  int inquire\_tele\_index(Tele t[], string name)*//从Tele[]中根据姓名查询Tele所在下标，返回位置*  {  *for* (int i = 0; i < Tele\_size; i++)      {  *if* (name.compare(t[i].name) == 0)          {  *return* i;          }      }      cout << "fail to find " << name << endl;  *return* -1;*//查找失败返回-1*  }  Tele inquire\_tele(Tele t[], string name)*//从Tele[]中根据姓名查询Tele，返回查到的结构体*  {  *return* t[inquire\_tele\_index(t, name)];  }  void insert\_tele(Tele t[], Tele temp)*//将temp插入Tele[]中*  {      t[Tele\_size++] = temp;  }  void delete\_tele(Tele t[], string name)*//从Tele[]中根据姓名删除对应Tele*  {  *for*(int i=inquire\_tele\_index(t,name)+1;i<=Tele\_size;i++){*//后面前挪覆盖*          t[i-1]=t[i];      }  *// memset(&t[Tele\_size-1], 0, sizeof(Tele));*      Tele\_size--;  }  void modi\_tele(Tele t[], int choice, string name, string info)  {*//根据choice和提供的name选择性修改Tele中的某项内容,choice:1、name 2、phoneNumber 3、mobileNumber 4、email*  *switch* (choice)*//根据choice选择功能*      {  *case* 1:          t[inquire\_tele\_index(t, name)].name = info;  *break*;  *case* 2:          t[inquire\_tele\_index(t, name)].phoneNumber = info;  *break*;  *case* 3:          t[inquire\_tele\_index(t, name)].mobileNumber = info;  *break*;  *case* 4:          t[inquire\_tele\_index(t, name)].email = info;  *break*;  *default*:          cout << "illegal choice" << endl;  *break*;      }  }  void print\_tele(Tele t)*//打印结构体*  {      cout << t.name << " " << t.phoneNumber << " " << t.mobileNumber << " " << t.email << endl;  }  void print\_teles(Tele t[])*//打印结构体数组*  {      cout<<"Tele数组中的内容为："<<endl;  *for* (int i = 0; i < Tele\_size; i++)*//遍历*      {          print\_tele(t[i]);      }  }  int main()*//测试*  {  *//插入*      Tele temp1={"小刚","111","222","333"};      insert\_tele(t,temp1);      print\_teles(t);      Tele temp2={"小明","444","555","666"};      insert\_tele(t,temp2);      print\_teles(t);  *//查询*      cout<<"以"<<temp1.name<<"为名的查询结果为："<<endl;      print\_tele(inquire\_tele(t,temp1.name));  *//删除*      cout<<"删除后还剩："<<endl;      delete\_tele(t,temp1.name);      print\_teles(t);  *//修改*      cout<<"修改结果："<<endl;      modi\_tele(t,4,temp2.name,"777");      print\_teles(t);  *//存入文件*      store\_tele\_to\_file(t);      cout<<"删除后还剩："<<endl;      delete\_tele(t,temp2.name);      print\_teles(t);  *//从文件读出*      extract\_tele\_from\_file(t);      print\_teles(t);  *return* 0;  } |

* 综合性实验

4.调试分析

1. 采用IDE中自带的调试功能进行调试，手动添加断点和查看程序。
2. 对设计和编码的讨论和分析。该程序实现了顺序栈的操作。分析程序代码的质量，主要从以下几个方面考虑。

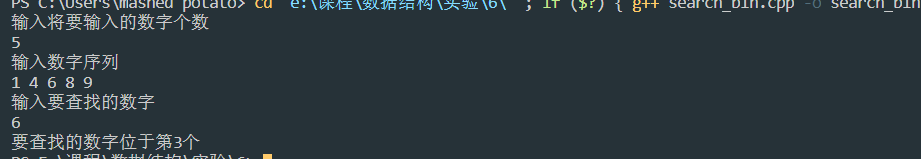
* 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。
* 健壮性。在一定的数据输入范围内，该程序能较好的实现操作。但是如果输入数 据非法，该程序还是可能会产生一些预想不到的输出结构，或是不做任何处理。所以， 该程序的健壮性有待进一步的提高。要综合考虑一些情况，当输入有误时，应返回一个 表示错误的值，并中止程序的执行，以便在更高的抽象层次上进行处理。

5.使用说明

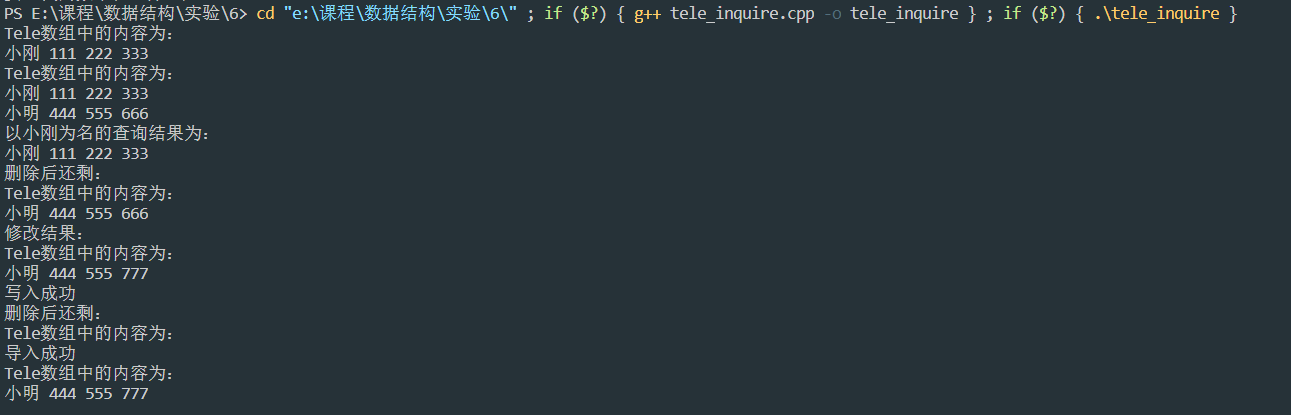
按照屏幕提示，选择想要的功能并输入对应数字，按下ENTER键后，根据屏幕提示进行输入，即可得到想要的结果。

6.测试程序运行结果

* 设计性实验



* 综合性实验



1. **心得体会**

通过本次实验，使我对数据结构有了更深的理解，对指针的运用更加熟练，熟悉了对函数的定义和操作。