山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201720130142 | 姓名： 黄孔进 | | 班级： 3班 |
| 实验题目：实验三 数组描述线性表 | | | |
| 实验学时：2 | | 实验日期： 2018 10 14 | |
| 实验目的：1、掌握线性表结构、数组描述方法（顺序存储结构）、数组描述线性表的实现。  2、掌握线性表应用。 | | | |
| 软件环境：  编辑器：Dev C++;  操作系统Windows 10 x64； | | | |
| 一、实验内容  1、创建线性表类：线性表的存储结构使用数组描述，提供操作: 插入、删除、  查找等。  2、设通讯录中每一个联系人的内容有：姓名、电话号码、班级、宿舍。由键  盘输入或文件录入的通讯录信息建立通讯录表，使用线性表中操作实现  通讯录管理功能，包括：插入、删除、编辑、查找（按姓名查找）；键盘  输入一班级，输出通讯录中该班级所有人信息。    二、数据结构与算法描述（整体思路描述，所需要的数据结构与算法）  数据结构：数组形式的线性表  算法：顺序查找  测试结果（测试输入，测试输出，结果分析）  测试输入：  操作2  Test.txt  操作1  Kevin 17860768809 3 B112  操作4  Kevin 17860768809 3 B111  操作6 3  操作3 Kevin  操作5 Kevin  QQ截图20181014184121QQ截图20181014184207QQ截图20181014184253QQ截图20181014184046QQ截图20181014184413QQ截图20181014184451  四、分析与探讨（结果分析，若存在问题，探讨解决问题的途径）  操作6：成功返回3班所有成员信息。  操作5：找不到Kevin，所以成功实现删除功能。  五、附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）  Exp3:  **Exp3.cpp：**  #include<iostream>  #include<cstring>  #include<string>  #include"addressList.h"  using namespace std;  void printContent(){  cout<<"=========通讯录========\n"  <<"1.insert by keyboard\n"  <<"2.insert by file\n"  <<"3.delete\n"  <<"4.edit\n"  <<"5.search\n"  <<"6.output a class\n"  <<"0.exit and save\n"  <<"========================\n"  <<"Your choice:";  }  void wait(){  cout<<endl<<"Enter to continue"<<endl;  getchar();getchar();  }  int main(){  addressList a;  int choice,cls;  string name,tel,dom;  do{  system("Cls");  printContent();  cin >> choice;  switch (choice){  case 1:  cout<<"Please input(name,telphone,class,dormitory)to insert\n:";  cin>>name>>tel>>cls>>dom;  a.keyboardInsert(name,tel,cls,dom);  wait();break;  case 2:  cout<<"Please input file's name to insert\n:";  cin>>name;  a.fileInsert(name);  wait();break;  case 3:  cout<<"Please input the name to delete\n:";  cin>>name;  a.erasion(name);  wait();break;  case 4:  cout<<"Please input (name,telphone,class,dormitory) to edit\n:";  cin>>name>>tel>>cls>>dom;  a.edit(name,tel,cls,dom);  wait();break;  case 5:  cout<<"Please input the name to search\n:";  cin>>name;  a.search(name);  wait();break;  case 6:  cout<<"Please input the class number to output\n:";  cin>>cls;  a.class\_output(cls);  wait();break;  case 0: a.save();break;  default: cout<<"Error choice!\n";wait();  }  } while(choice != 0);  return 0;  }  **addressList.h：**  #ifndef addressList\_  #define addressList\_  #include<iostream>  #include<fstream>  #include<cstring>  #include<string>  #include<iomanip>  #include"arrayList.h"  using namespace std;  class addressList{  arrayList<string> \_name;  arrayList<string> \_tel;  arrayList<int> \_cls;  arrayList<string> \_dom;  int num;  public:  addressList(){num=0;}  void save();  void keyboardInsert(string name,string tel,int cls,string dom);  void fileInsert(string file);  void search(string name);  void erasion(string name);  void edit(string name,string tel,int cls,string dom);  void class\_output(int cls);  };  void addressList::save(){//保存信息  ofstream fout("Exp3.txt");  if(!fout){cout<<"file open failure\n";}  else{  fout<<num<<endl;  for (int i = 0;i < num;i++){  fout<<\_name.get(i) << endl;  fout<<\_tel.get(i) << endl;  fout<<\_cls.get(i) << endl;  fout<<\_dom.get(i) << endl;  }  }  fout.close();  }  void addressList::keyboardInsert(string name,string tel,int cls,string dom){//用键盘插入信息  \_name.insert(num,name);  \_tel.insert(num,tel);  \_cls.insert(num,cls);  \_dom.insert(num,dom);  num++;  }  void addressList::fileInsert(string file){//用文件插入信息  const char\* ch;  ch = file.c\_str();  ifstream fin(ch);  if(!fin){cout<<"file open failure\n";}  else{  string name,tel,dom;  int cls;  fin>>num;  for (int i = 0;i < num;i++){  fin>>name>>tel>>cls>>dom;  \_name.insert(i,name);  \_tel.insert(i,tel);  \_cls.insert(i,cls);  \_dom.insert(i,dom);  }  }  fin.close();  }  void addressList::search(string name){ //以姓名为索引，查找信息  int id=\_name.indexOf(name);  if(id==-1){cout<<"No one called:"+name<<endl;}  else{  cout<<"name:"<<\_name.get(id)<<endl  <<"telphone:"<<\_tel.get(id)<<endl  <<"class:"<<\_cls.get(id)<<endl  <<"domitory:"<<\_dom.get(id)<<endl;  }  }  void addressList::erasion(string name){//以姓名为索引，删除信息  int id=\_name.indexOf(name);  if(id==-1){cout<<"No one called:"+name<<endl;}  else{  \_name.erase(id);  \_tel.erase(id);  \_cls.erase(id);  \_dom.erase(id);  num--;  }  }  void addressList::edit(string name,string tel,int cls,string dom){  //以姓名为索引，编辑信息  int id=\_name.indexOf(name);  if(id==-1){cout<<"No one called:"+name<<endl;}  else{  \_tel.erase(id);  \_cls.erase(id);  \_dom.erase(id);  \_tel.insert(id,tel);  \_cls.insert(id,cls);  \_dom.insert(id,dom);  }  }  void addressList::class\_output(int cls){//输出通讯录中班级cls所有人信息  cout<<"class"<<cls<<" information:"<<endl;  cout<<"name"<<setw(15)<<"telphone"<<setw(15)<<"dormitory"<<endl;  for(int i=0;i<num;i++){  if(\_cls.get(i)==cls)  cout<<setw(6)<<\_name.get(i)<<setw(15)  <<\_tel.get(i)<<setw(10)  <<\_dom.get(i)<<endl;  }  }  #endif  **arrayList.h：**  #ifndef arrayList\_  #define arrayList\_  #include<iostream>  #include<cstdio>  #include<sstream>  #include<iterator>  #include<algorithm>  #include<cstring>  #include<string>  using namespace std;  class illegalParameterValue {  public:  illegalParameterValue(string message = "Illegal parameter value"){  cout << message << endl;  }  };  template<class T>  class arrayList{  public:  //构造、复制构造和析构函数  arrayList(int initialCapacity=10);  arrayList(const arrayList<T>&);  ~arrayList(){delete[] element;}    //ADT方法  bool empty()const{return listSize==0;}  int size()const {return listSize;}  T& get(int theIndex) const;  int indexOf(const T& theElement)const;  void erase(int theIndex);  void insert(int theIndex,const T& theElement);  void output(ostream& out)const;    //其他方法  int capacity()const{return arrayLength;}  protected:  void checkIndex(int theIndex)const;  T\* element;  int arrayLength;  int listSize;  };  template<class T>  arrayList<T>::arrayList(int initialCapacity){  if (initialCapacity<1){  ostringstream s;  s<<"Initial capacity ="<<initialCapacity<<" Must be > 0";  throw illegalParameterValue(s.str());  }  arrayLength=initialCapacity;  element=new T[arrayLength];  listSize=0;  }  template<class T>  arrayList<T>::arrayList(const arrayList<T>& theList){  arrayLength=theList.arrayLength;  listSize=theList.listSize;  element=new T[arrayLength];  copy(theList.element,theList.element+listSize,element);  }  template<class T>  void arrayList<T>::checkIndex(int theIndex)const {  if(theIndex<0||theIndex>=listSize){  ostringstream s;  s<<"index = "<<theIndex<<",but size = "<<listSize;  throw illegalParameterValue(s.str());  }  }  template<class T>  T& arrayList<T>::get(int theIndex)const {//返回索引为theIndex的元素  checkIndex(theIndex);  return element[theIndex];  }  template<class T>  int arrayList<T>::indexOf(const T& theElement)const {  //返回元素theElement第一次出现时的索引  int theIndex=(int)(find(element,element+listSize,theElement)-element);  if(theIndex==listSize)return -1;  else return theIndex;  }  template<class T>  void changeLength1D(T\* &a,int oldLength,int newLength){  if(newLength<0)throw illegalParameterValue("Error:new length must be>0");  T\* temp=new T[newLength];  int number=min(oldLength,newLength);  copy(a,a+number,temp);  delete[]a;  a=temp;  }  template<class T>  void arrayList<T>::erase(int theIndex){//删除索引为theIndex的元素  checkIndex(theIndex);  if(listSize\*4<=arrayLength){  changeLength1D(element,arrayLength,arrayLength/2);  arrayLength/=2;  }  copy(element+theIndex+1,element+listSize,element+theIndex);  element[--listSize].~T();  }  template<class T>  void arrayList<T>::insert(int theIndex,const T&theElement){  //把theElement插入线性表中索引为theIndex的位置上  if(theIndex<0||theIndex>listSize){  ostringstream s;  s<<"index = "<<theIndex<<"but size = "<<listSize;  "Error:new length must be>0";  }  if(listSize==arrayLength){  changeLength1D(element,arrayLength,2\*arrayLength);  arrayLength\*=2;  }  copy\_backward(element+theIndex,element+listSize,element+listSize+1);  element[theIndex]=theElement;  listSize++;  }  template<class T>  void arrayList<T>::output(ostream& out)const {//把线性表插入输出流out  copy(element,element+listSize,ostream\_iterator<T>(cout," "));  }  template<class T>  ostream& operator<<(ostream& out,const arrayList<T>& x){  x.output(out);  return out;  }  #endif | | | |