**数据结构课程设计报告**

**班 级： 计科124**

**姓 名：王腾涛（20121515105）**

**王坤龙（20121515104 ）**

**赵振伟（20121515102）**

**日 期： 2014/10/30**

**通讯录管理系统**

**一、问题描述**

基本要求如下：

（1）用顺序表作为通讯录的存储结构。每条通讯录包含：姓名，城市，电话。表中的记录按姓名非递减有序，每插入一条记录后，应使表中记录仍保持按姓名非递减排序。

（2）实现插入、删除、修改、显示、查找功能（插入、删除、查找均以姓名作为关键字折半查找）

**二、概要设计**

**1，**抽象数据类型线性表的定义如下：

**ADT list**{

数据对象：D={ai|ai ϵElemSet,i=1,2,...,n ,n≥0}

数据关系：R1={<a i-1,ai>|a i-1,aiϵD,i=2,...,n}

基本操作：

InitList(&L)

操作结果：构造一个空的线性表。

PriorElem（L,cur\_e,&pre\_e）

初始条件：线性表L已存在（可以为空）。

操作结果：若cur\_e是L的数据元素，则用pre\_e返回它的前驱（若只有一个元素，则返回其自身），否则操作失败，pre\_e无定义。

NextElem（L,cur\_e,&next\_e）

初始条件：线性表L已存在（可以为空）。

操作结果：若cur\_e是L的数据元素，则用next\_e返回它的后继（若只有一个元素，则返回其自身），否则操作失败，pre\_e无定义。

ListInsert(&L,i,e)

初始条件：线性表L已存在，1≤i≤ListLength(L)+1。

操作结果：在L中第I个位置之前插入新的数据元素e，L的长度加1。

ListDelete(&L,i,&e)

初始条件：线性表L已存在且非空，1≤i≤ListLength(L)。

操作结果：删除L的第i个数据元素，并用e返回其值，L的长度减1。

}ADT List

2，本程序包含3个模块：

1. 主程序模块：

Void main() {

初始化；

while(条件){

接受命令；

处理命令；

};

}

1. 顺序表模块——实现顺序表的抽象数据类型及对其的基本操作；
2. 类模块——实现类对象中基本数据的存储。

各模块之间的调用关系大体如下：

主程序模块

顺序表模块

类模块

**三、详细设计**

1，元素类型，结点类型和指针类型

//定义节点类型

#define LIST\_INIT\_SIZE 100 //存储空间初始分配量

#define LISTINCREMENT 10 //存储空间的分配增量

typedef struct DulNode

{

ElemType data[100];

int length;

}DulNode;

2，主要算法如下：

1）//排序

void nodeSort(DulNode &list,int local,string oldName,string newName)

{

int low = 0;

int high = 0;

int mid = 0;

int temp = 0;

if(newName.compare(oldName) <= 0)

{

low = 0;

high = local - 1;

temp = 1; //向左查找

}else

{

low = local + 1;

high = list.length-1;

temp = 0; //向右查找

}

while(low <= high)

{

mid = (low + high) / 2;

if(newName.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

Person tempPerson = list.data[local];

if(temp == 1) //向右移动

{

for(int j = local - 1 ;j >= high + 1;--j)

{

list.data[j + 1] = list.data[j];

}

list.data[(high + 1)] = tempPerson;

}else

{

for(int j = local + 1 ;j <= high; j++)

{

list.data[j - 1] = list.data[j];

}

list.data[(high)] = tempPerson;

}

}

2）//插入节点,折半插入

Status nodeInsert(DulNode &list, Person person)

{

int len = list.length-1;

if(len >= 0)

{

int low = 0; int high = len;

int mid = 0;

while(low <= high)

{

mid = (low + high) / 2;

if(person.getName().compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

for(int j = len;j >= high + 1;--j)

{

list.data[j + 1] = list.data[j];

}

list.data[(high + 1)] = person;

}

else

{

list.data[0] = person;

}

list.length++;

return SUCCESS;

}

3）//删除节点

Status nodeDelete(DulNode &list, string name)

{

if(list.length > 0)

{

int low = 0;

int high = list.length-1;

int mid = (low + high) / 2;

while(low <= high)

{

mid = (low + high) / 2;

if(name.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else if(name.compare(list.data[mid].getName()) > 0)

{

low = mid + 1;

}

else

{

for(int i = mid + 1;i <= list.length - 1;i ++)

{

list.data[i - 1] = list.data[i];

}

list.length --;

return SUCCESS;

}

}

return ERROR;

}

return ERROR;

}

4）//判断用户是否存在并获取该节点的数据

Status isNode(DulNode &list, string name, Person\*\* person)

{

if(list.length > 0)

{

int low = 0;

int high = list.length-1;

int mid = (low + high) / 2;

while(low <= high)

{

mid = (low + high) / 2;

if(name.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

if(name.compare(list.data[high].getName()) == 0)

{

\*person = &(list.data[high]);

return high;

}

}

return ERROR;

}

由主函数调用各个操作函数，例如插入，删除，查询等函数，而各个操作函数又会对顺序表操作进行调用，对象的基本数据在Person类中。

**四、调试分析**

1：数据结构初始化之后，内存分配失败

解决：不使用malloc()函数进行动态的分配内存，把存储空间指定为100，问题解决但是效果不好；

2：数据插入之后，不是按照非递减顺序排序的

解决：使用折半查找找到指定的位置，这样直接插入到指定的位置后，数据就是有序的；

3：删除一个数据后，发现显示时仍然可以显示

解决：删除操作没有问题，显示方法的逻辑有问题；

4：修改一个指定的数据后，发现全部数据没有排序

解决：定义一个排序方法（折半插入排序），在修改之后，对其进行再次排序。

**五、心得体会**

本次课程设计，我们三人一组，基本分工如下：首先由我们三人共同探讨问题的各种情况，做一个简单的需求分析。

王腾涛主要负责整体框架的架构以及查询函数的编写，

王坤龙主要负责人机交互界面的编写及该系统的测试工作，

赵振伟主要负责该系统的调试以及本设计文档的撰写。

心得与感悟：虽然只是一次简单的课程设计，但还是发现掌握在自己手里的知识并不可靠，很多以前学过的知识都变的混淆，通过不断的从书上，网上查资料来解决。在程序设计初期，由于事先没有沟通好，我们三个人出现了较大的分歧，致使延缓了程序的完成，所幸我们重新进行了整个系统的设计与分析，把思路搞清，顺利完成程序的设计。从中我了解到，在以后的工作中我们会经常遇到合作性的工作，这时我们一定要事先就沟通好，这样在任务的完成的过程中，才能最大的避免失误而引起的误工。

**六、用户使用说明**

1)本程序的运行环境为Microsoft Visual C++ 6.0,程序的运行文件为address\_list.cpp;

2)打开运行后，会进入通讯录的欢迎界面，在界面上会有

（1）插入（2）查询（3）修改（4）删除（5）显示全部

（6）退出程序；六个菜单。

通过界面的提示按键来执行，在执行期间，如果想返回主界面，可以直接按（q）键退出。

**七、测试结果**

（省略）

**八、附源码清单如下：**

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<string>

using namespace std;

/\* person类

包含人的姓名，城市，电话

\*/

class Person

{

private:

string name;

string city;

string phone;

public:

Person(){}

Person(string name, string city, string phone)

{

this->name = name;

this->city = city;

this->phone = phone;

}

string getCity()

{

return this->city;

}

void setCity(string city)

{

this->city = city;

}

string getPhone()

{

return this->phone;

}

void setPhone(string phone)

{

this->phone = phone;

}

string getName()

{

return this->name;

}

void setName(string name)

{

this->name = name;

}

void update(string name, string city, string phone)

{

this->name = name;

this->city = city;

this->phone = phone;

}

};

typedef Person ElemType;

typedef int Status;

const int SUCCESS = 1;

const int ERROR = -2;

//定义节点类型

typedef struct DulNode

{

ElemType data[100];

int length;

}DulNode;

//排序

void nodeSort(DulNode &list,int local,string oldName,string newName)

{

int low = 0;

int high = 0;

int mid = 0;

int temp = 0;

if(newName.compare(oldName) <= 0)

{

low = 0;

high = local - 1;

temp = 1; //向左查找

}else

{

low = local + 1;

high = list.length-1;

temp = 0; //向右查找

}

while(low <= high)

{

mid = (low + high) / 2;

if(newName.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

Person tempPerson = list.data[local];

if(temp == 1) //向右移动

{

for(int j = local - 1 ;j >= high + 1;--j)

{

list.data[j + 1] = list.data[j];

}

list.data[(high + 1)] = tempPerson;

}else

{

for(int j = local + 1 ;j <= high; j++)

{

list.data[j - 1] = list.data[j];

}

list.data[(high)] = tempPerson;

}

}

//插入节点,折半插入

Status nodeInsert(DulNode &list, Person person)

{

int len = list.length-1;

if(len >= 0)

{

int low = 0; int high = len;

int mid = 0;

while(low <= high)

{

mid = (low + high) / 2;

if(person.getName().compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

for(int j = len;j >= high + 1;--j)

{

list.data[j + 1] = list.data[j];

}

list.data[(high + 1)] = person;

}

else

{

list.data[0] = person;

}

list.length++;

return SUCCESS;

}

//删除节点

Status nodeDelete(DulNode &list, string name)

{

if(list.length > 0)

{

int low = 0;

int high = list.length-1;

int mid = (low + high) / 2;

while(low <= high)

{

mid = (low + high) / 2;

if(name.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else if(name.compare(list.data[mid].getName()) > 0)

{

low = mid + 1;

}

else

{

for(int i = mid + 1;i <= list.length - 1;i ++)

{

list.data[i - 1] = list.data[i];

}

list.length --;

return SUCCESS;

}

}

return ERROR;

}

return ERROR;

}

//判断用户是否存在并获取该节点的数据(修改)

Status isNode(DulNode &list, string name, Person\*\* person)

{

if(list.length > 0)

{

int low = 0;

int high = list.length-1;

int mid = (low + high) / 2;

while(low <= high)

{

mid = (low + high) / 2;

if(name.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else

{

low = mid + 1;

}

}

if(name.compare(list.data[high].getName()) == 0)

{

\*person = &(list.data[high]);

return high;

}

}

return ERROR;

}

//根据姓名判断用户是否存在并获取该节点的数据（查找用户）

Status selectNode(DulNode &list, string name, Person\*\* person)

{

if(list.length > 0)

{

int low = 0;

int high = list.length-1;

int mid = (low + high) / 2;

while(low <= high)

{

mid = (low + high) / 2;

if(name.compare(list.data[mid].getName()) < 0)

{

high = mid - 1;

}else if(name.compare(list.data[mid].getName()) > 0)

{

low = mid + 1;

}else

{

\*person = &(list.data[mid]);

return SUCCESS;

}

}

}

return ERROR;

}

//主菜单

void showMenu()

{

printf("\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 欢迎使用通讯录管理系统 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* \*\n");

printf("\* 　 　1、添加通讯录　");

printf(" 　 　 2、查询　　 　\*\n");

printf("\* 　　　　　　　　3、修改 ");

printf(" 　　　 4、删除记录　　　　 \*\n");

printf("\* 　　　　　　　　5、显示全部 ");

printf(" 　　　 6、退出程序 　 \*\n");

printf("\* \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

//添加新成员

int addPerson(DulNode &list)

{

char name[20];

char city[20];

char phone[20];

printf("请输入姓名【可输入q回到初始界面操作】：");

scanf("%s", name);

fflush(stdin);

string sName = (string)name;

if(sName == "q")

{

printf("\n");

return 0;

}

printf("\n请输入城市：");

scanf("%s", city);

printf("\n请输入电话：");

scanf("%s", phone);

Person \*person = new Person(sName, city, phone);

nodeInsert(list, \*person);

printf("\n添加成功！\n\n");

return 1;

}

//显示所有成员

void showAllPerson(DulNode &list)

{

printf("\n=========================== 通讯录列表 ===========================\n\n");

int len = list.length;

for(int j = 0;j <= list.length - 1; j ++)

{

string name = list.data[j].getName();

string city = list.data[j].getCity();

string phone = list.data[j].getPhone();

printf("\* 编号：%d 姓名：%s 城市：%s 电话：%s\n\n",j + 1, name.c\_str(), city.c\_str(),phone.c\_str());

}

if(len == 0)

{

printf(" 数据为空，请添加数据\n\n");

}

printf("==================================================================\n\n");

}

//修改用户信息

void updatePerson(DulNode &list)

{

char inName[20];

printf("请输入需要修改用户的姓名 【可输入q回到初始界面操作】：");

scanf("%s", &inName);

fflush(stdin);

string sName = (string)inName;

if(sName == "q")

{

printf("\n");

return;

}

Person\* person = new Person();

int high = isNode(list, sName, &person);

while(high == -2)

{

printf("用户名不存在！ \n\n");

printf("请输入需要修改用户的姓名 【可输入q回到初始界面操作】：");

char temp[20]; //临时变量

scanf("%s", &temp);

fflush(stdin);

sName = (string)temp;

if(sName == "q")

{

printf("\n");

return;

}

high = isNode(list, sName, &person);

}

string oldName = list.data[high].getName();

char name[20];

char city[20];

char phone[20];

printf("请输入新的用户信息\n\n");

printf("请输入姓名：");

scanf("%s", &name);

printf("\n请输入城市：");

scanf("%s", &city);

printf("\n请输入电话：");

scanf("%s", &phone);

person->update(name, city, phone);

printf("\n修改成功\n\n");

string newName = (string)name;

nodeSort(list,high,oldName,newName);

showAllPerson(list);

}

//查询成员信息

void selectPerson(DulNode &list)

{

char inName[20];

printf("请输入需要查询用户的姓名 【可输入q回到初始界面操作】：");

scanf("%s", &inName);

fflush(stdin);

string sName = (string)inName;

if(sName == "q")

{

printf("\n");

return;

}

Person \*person = new Person();

while(selectNode(list, sName, &person) == -2)

{

printf("用户名不存在！\n\n");

printf("请再次输入要查询用户的姓名【可输入q回到初始界面】：");

char temp[20]; //临时变量

scanf("%s", &temp);

fflush(stdin);

sName = (string)temp;

if(sName == "q")

{

printf("\n");

return;

}

}

string name = person->getName();

string city = person->getCity();

string phone = person->getPhone();

printf("\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");

printf("\* \*\n");

printf("\* 　 　姓名：%s　　 　\n\n",name.c\_str());

printf("\* 　 　城市：%s ",city.c\_str());

printf(" 电话：%s　　 　\n",phone.c\_str());

printf("\* \*\n");

printf("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n\n");

}

//删除成员信息

void deletePerson(DulNode &list)

{

char inName[20];

printf("请输入需要删除用户的姓名【可输入q放弃删除操作】：");

scanf("%s", &inName);

fflush(stdin);

string sName = (string)inName;

if(sName == "q")

{

printf("\n");

return;

}

while(nodeDelete(list, sName) == -2)

{

printf("用户姓名不正确\n\n");

printf("请再次输入需要删除用户的姓名【可输入q放弃删除操作】：");

char temp[20]; //临时变量

scanf("%s", &temp);

fflush(stdin);

sName = (string)temp;

if(sName == "q")

{

printf("\n");

return;

}

}

printf("删除成功！\n\n");

}

//主函数

int main()

{

DulNode L;

L.length = 0;

int mainChance;

showMenu();

printf("请输入:");

scanf("%d", &mainChance);

fflush(stdin);

int temp = 0;

while(mainChance!=6)

{

switch(mainChance)

{

case 1:

temp = addPerson(L);

if(temp == 1)

{

showAllPerson(L);

}

break;

case 2:

selectPerson(L);

break;

case 3:

updatePerson(L);

break;

case 4:

deletePerson(L);

break;

case 5:

showAllPerson(L);

break;

default:

printf("\n请输入 1 ~ 6 之间的整数 \n\n");

break;

}

mainChance = 0;

showMenu();

printf("请输入:");

scanf("%d", &mainChance);

fflush(stdin);

}

return 0;

}