

“程序设计基础”

课程设计报告

**设计题目**  简单的通讯录管理系统

**姓 名**  党存远

**学 号**  2022217587

**专 业**  物联网工程

**班 级**  物联网工程（2）班

**完成日期**  2023.3.10

|  |  |  |  |
| --- | --- | --- | --- |
| **“程序设计基础课程设计课程”验收细则** | | | |
| **成绩等级** | **具体表现** | **教师评分**  **（百分制）** | |
| A（100-85] | 1）能够在规定时间内完成所选项目，所完成的项目完整性较高，具有一定难度，交互良好；2）工作量较大；3）自学了超出课程之外的编程技术；4）能正确回答教师提问； | □ |  |
| B（85-75] | 1）能够在规定时间内完成项目，且项目具有较好的完整性，交互设计较为优秀，难度一般；2）工作量较为充足；3）能正确回答教师提问； | □ |  |
| C（75-65] | 1）能够在规定时间内完成项目，项目完整性较好，交互设计一般；2）工作量较小； | □ |  |
| D（65-60] | 1）虽完成项目编码但未能在规定时间内完成的； | □ |  |
| E（<60） | 1）未进行验收的； | □ |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **“程序设计基础课程设计课程”报告评分细则** | | | |
| **成绩等级** | **具体表现** | **教师评分（百分制）** | |
| A（100-85] | 1）课程设计报告格式完美，充分采用图表来说明问题，章节设计完整、优秀，工作量饱满，排版美观； | □ |  |
| B（85-75] | 1）课程设计报告格式良好，采用了图表来说明问题，章节设计良好，工作量较为饱满； | □ |  |
| C（75-65] | 1）课程设计报告格式较好，章节设计较好，工作量一般； | □ |  |
| D（65-60] | 1）虽提交报告但未能在规定时间内完成的； | □ |  |
| E（<60） | 1）未提交报告的； | □ |  |

教师签名：

目录

[1 需求分析和规格说明 1](#_Toc127868519)

[2 设计 1](#_Toc127868520)

[3 用户手册 15](#_Toc127868521)

[4 调试及测试 15](#_Toc127868522)

[5 运行实例 19](#_Toc127868523)

[6 进一步改进 19](#_Toc127868524)

[7 心得体会 19](#_Toc127868525)

[8 对课程设计的建议 20](#_Toc127868526)

[9 附录1 参考文献 20](#_Toc127868527)

[10 附录2⎯⎯源程序 20](#_Toc127868528)

## 1 需求分析和规格说明

设计实现一个简单的通讯录管理系统，对联系对象的固定电话、手机、邮箱、 QQ 好友等进行管理。

要求：

1. 能够添加、删除、修改通讯录中已有的记录；
2. 要求对通讯录可以实现分组管理的功能；
3. 需要自行设计存储文件，不能使用现有数据库管理系统。

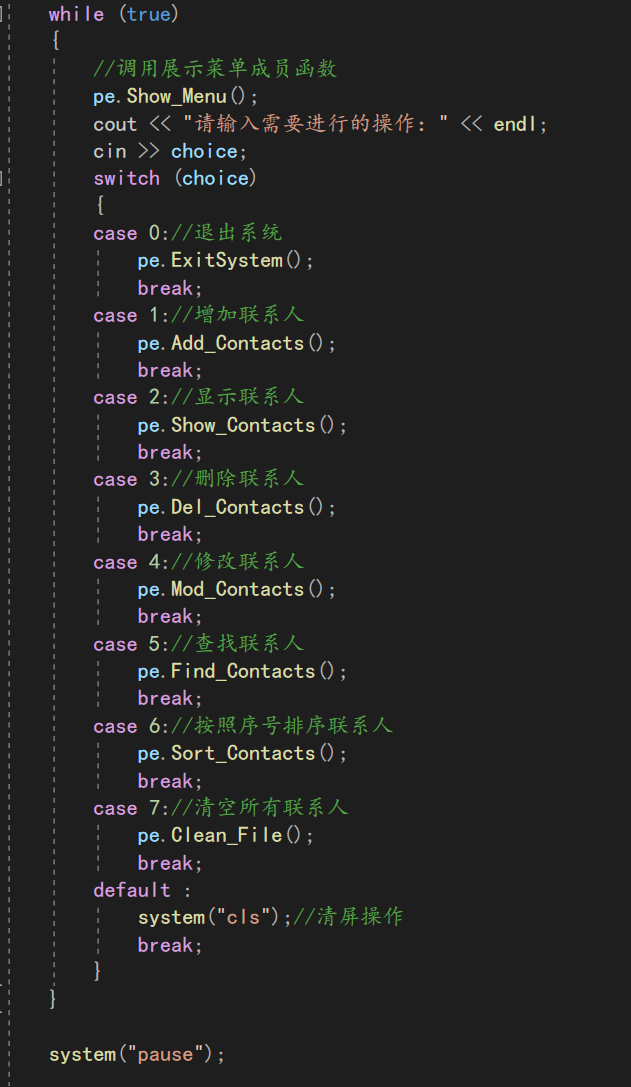
## 2 设计

根据上述需求，设计一个类Person（父类）代表所有联系人以及对于联系人的所有操作,然后在该类下重新设置一个包含所有联系人类型的People（父类），在该类下设置所有联系人将拥有的属性以及对联系人的各类信息进行打印输出，由People的基类派生出所有联系人分组的子类：Family（家人）类、Friend（朋友）类和Lover（情侣）类。三者都以公有的方式继承父类People。

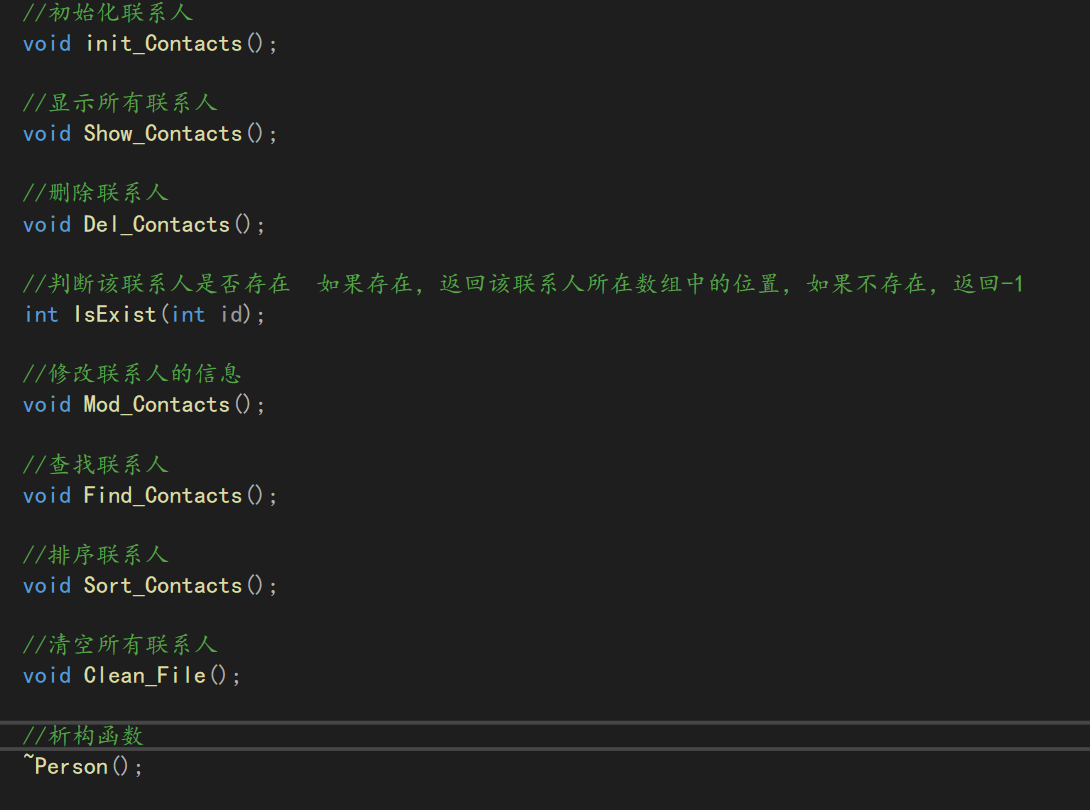
在Person基类中，除了定义构造函数和析构函数以外，还应统一定义对各类人员信息都应有的操作，这样可以规范各派生类的基本行为。

首先编写一个联系人管理系统的cpp文件，在该文件下包含Person.h(实现对于联系人的各种操作的头文件),People.h(一个联系人的虚类),Friend.h(朋友类),Family(家人类),Lover(亲密关系情侣类)。

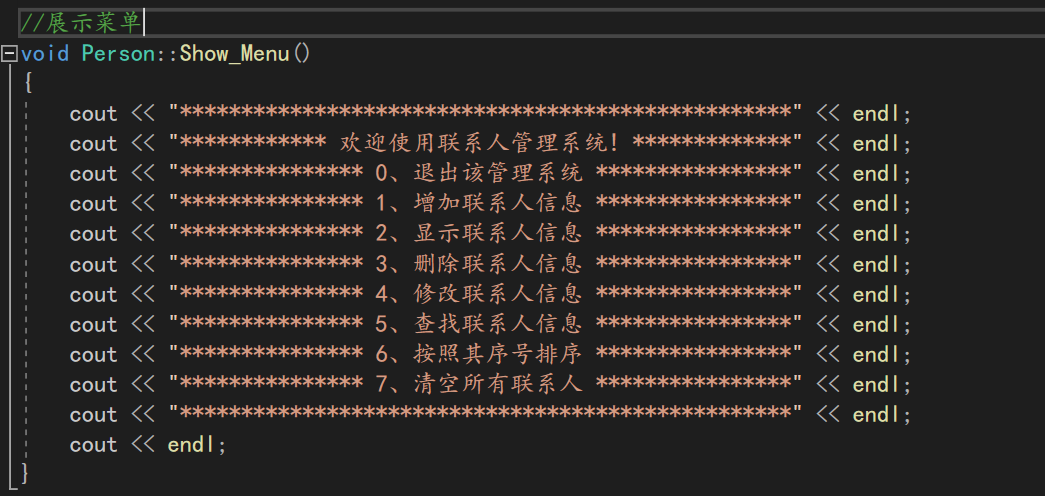
在联系人管理系统的cpp文件下实现让用户选择不同的选项，通过不同的选项对于联系人进行不同的操作，如图所示。



图一

在Person.h的文件中分别声明switch下的各类分支选项。

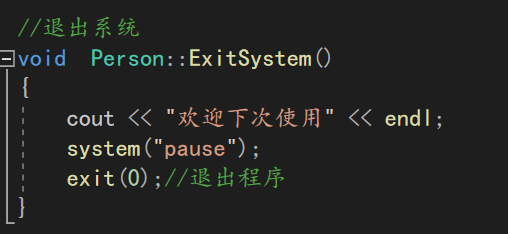
图二

首先在Person.cpp文件中实现菜单的展示，具体实现函数如下：

图三

在Person.cpp文件下实现0—7的所有操作如下：

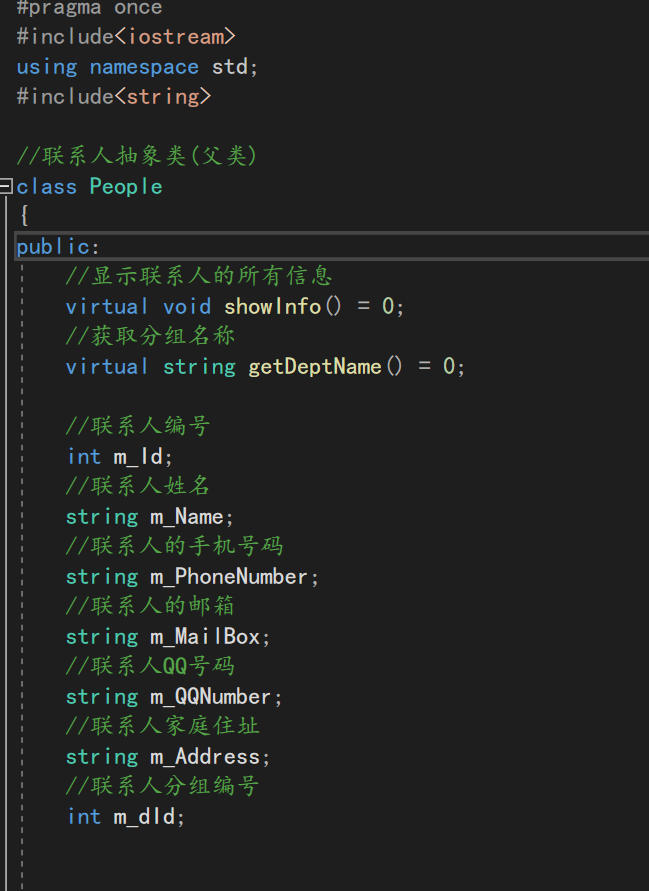
1. 封装一个函数实现对于该操作系统的退出。



图四

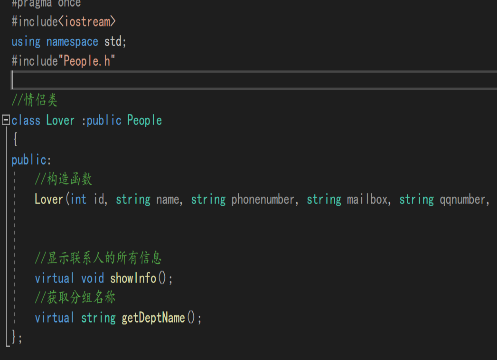
1. 增加联系人的信息。

a.首先创建职工的抽象类，其中包含一些关于联系人的基本属性与方法。如图所示：

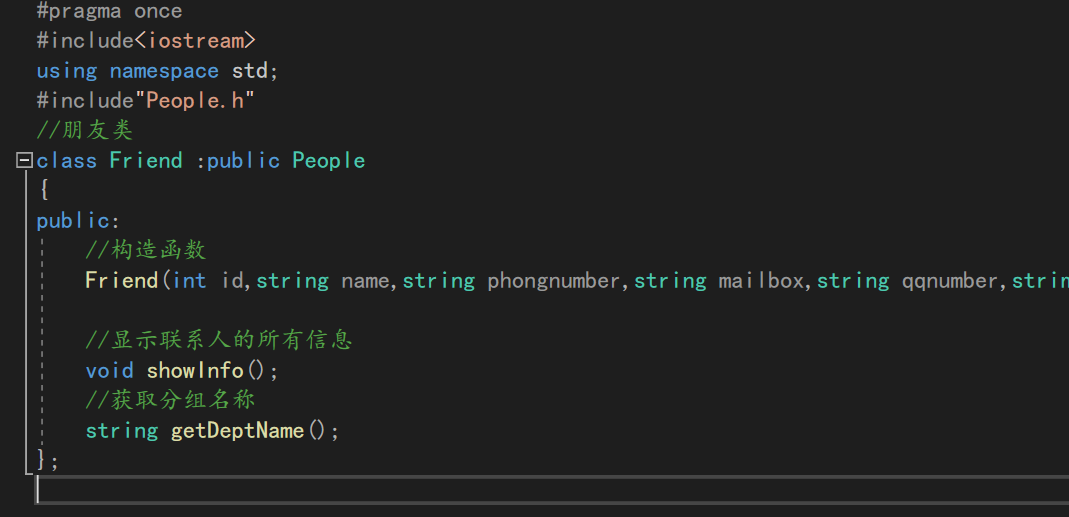


图五

b.在父类(Person)下衍生出三个子类分别为Friend(朋友类),Family(家人类),Lover(亲密关系情侣类)。

具体实现如下图所示：

图六



c.创建添加联系人函数(批量添加联系人，并且保存在文件中)

通过存放在堆区的联系人数组来存放各个联系人的信息。

注意：由于,不同的对象存放在数组中的数据类型不同，有Friend(朋友类),Family(家人类),Lover(亲密关系情侣类),因此为了让一个数组去存储三种不同类型的对象，此时使用这三个子类的父类People类型数据存放在数组中，之后通过父类的指针来指向子类的对象，因此需要创建一个数组中存放的是People\*类(父类)的指针，又因为需要将这个数组中的每个元素都存放在堆区，通过我们自己来实现对于这些数据的释放，因此需要创建一个People\*\*(二级指针)来申请一个在堆区存放的数组。

new Family new Friend new Lover



|  |  |  |
| --- | --- | --- |
| People\* | People\* | People\* |

People\*\*=new People[新空间大小]。

具体实现如下图所示：

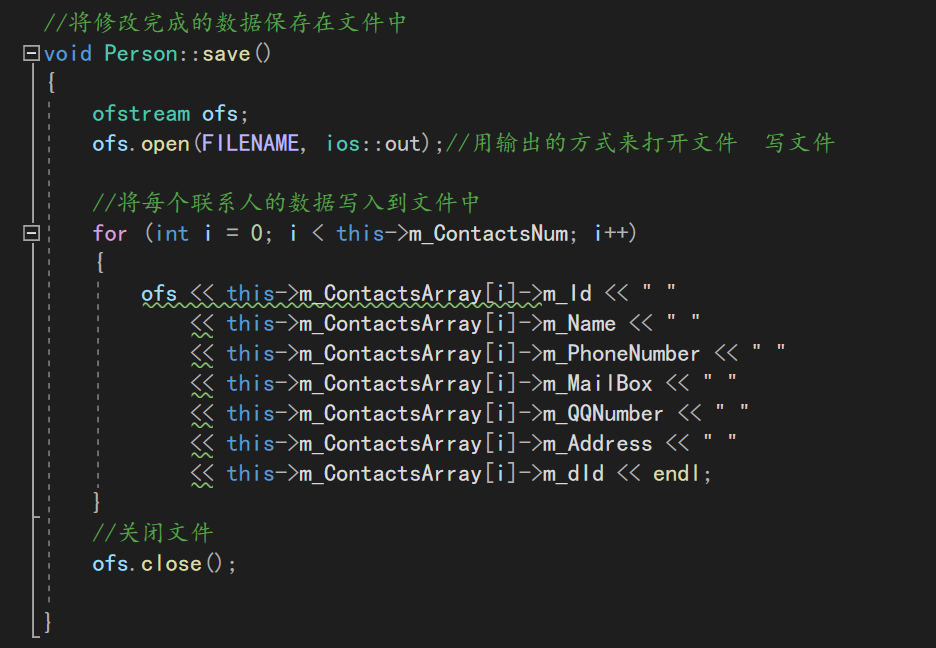


图七：

图八

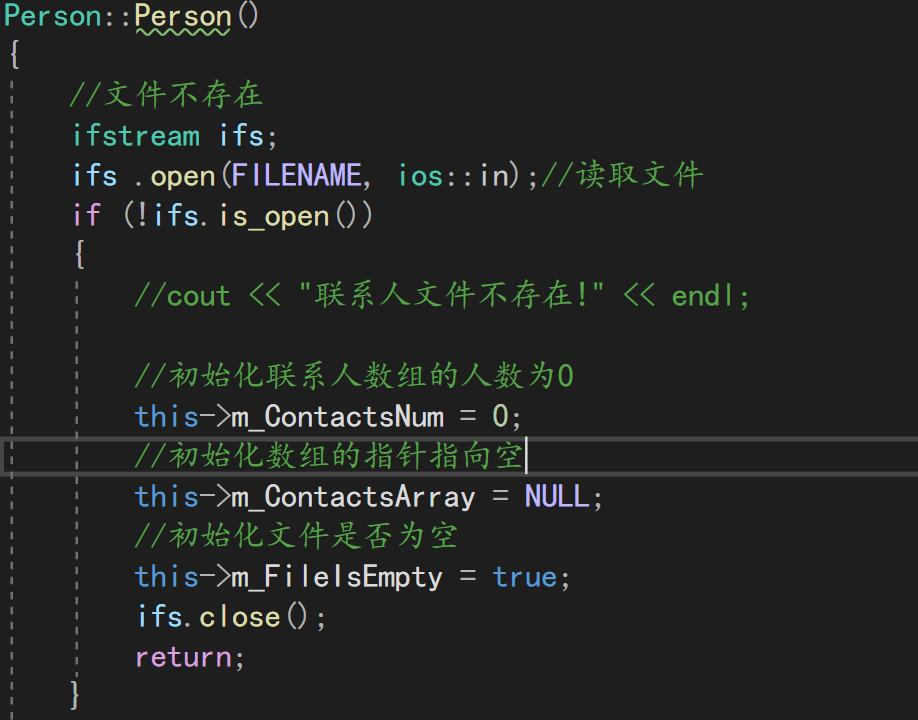
d.当将联系人的信息全部添加完毕之后，通过写文件的操作将所有的联系人信息保存在文件中，防止联系人信息的丢失。

具体实现如下图所示：

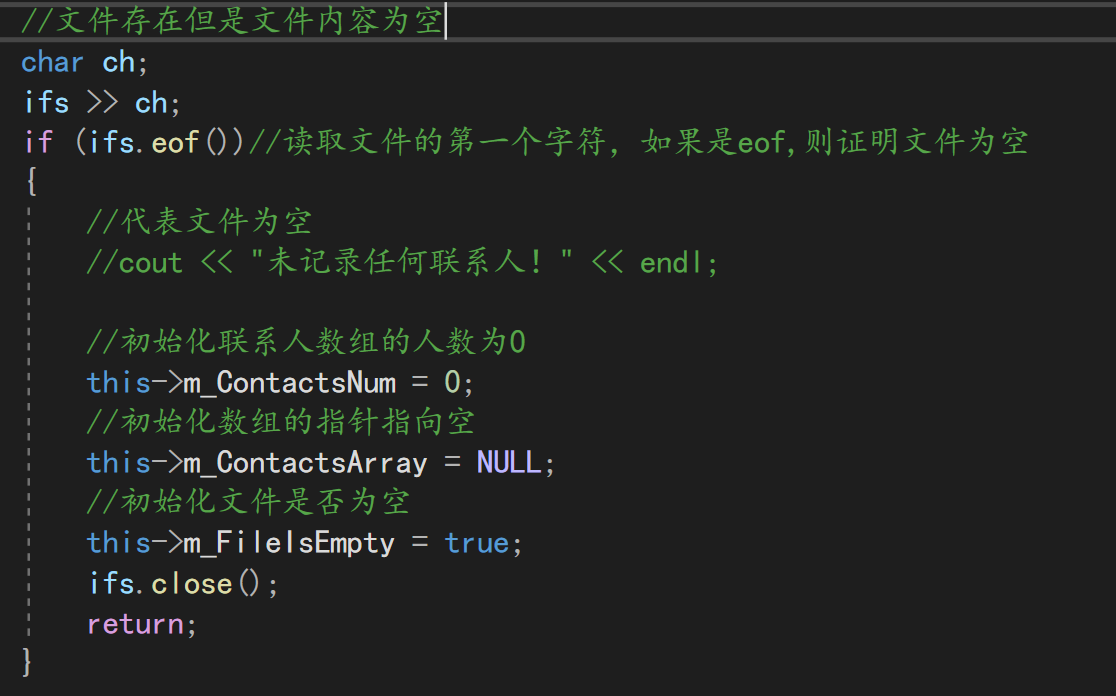
图九

f.在进行添加联系人时还要注意，有三种情况！Person的构造函数下，分三种情况对于联系人的信息进行储存。

1.当该文件在开始时还未进行创建时，创建一个新的文件存储在程序的运行文件下，命名为(contacts.txt)文件。之后对于存储联系人信息的数组进行初始化，如图一所示。



图十

2.当保存联系人所有的信息的文件存在时(contacts.txt文件存在)，但是文件内未存储任何联系人的信息时，通过第二个构造函数初始化。

图十一

3.最后一种即为最常见操作，此时文件存在并且已经存储了部分联系人信息的数据，此时对于对象的初始化操作如下

图十二

1. 显示职工操作。

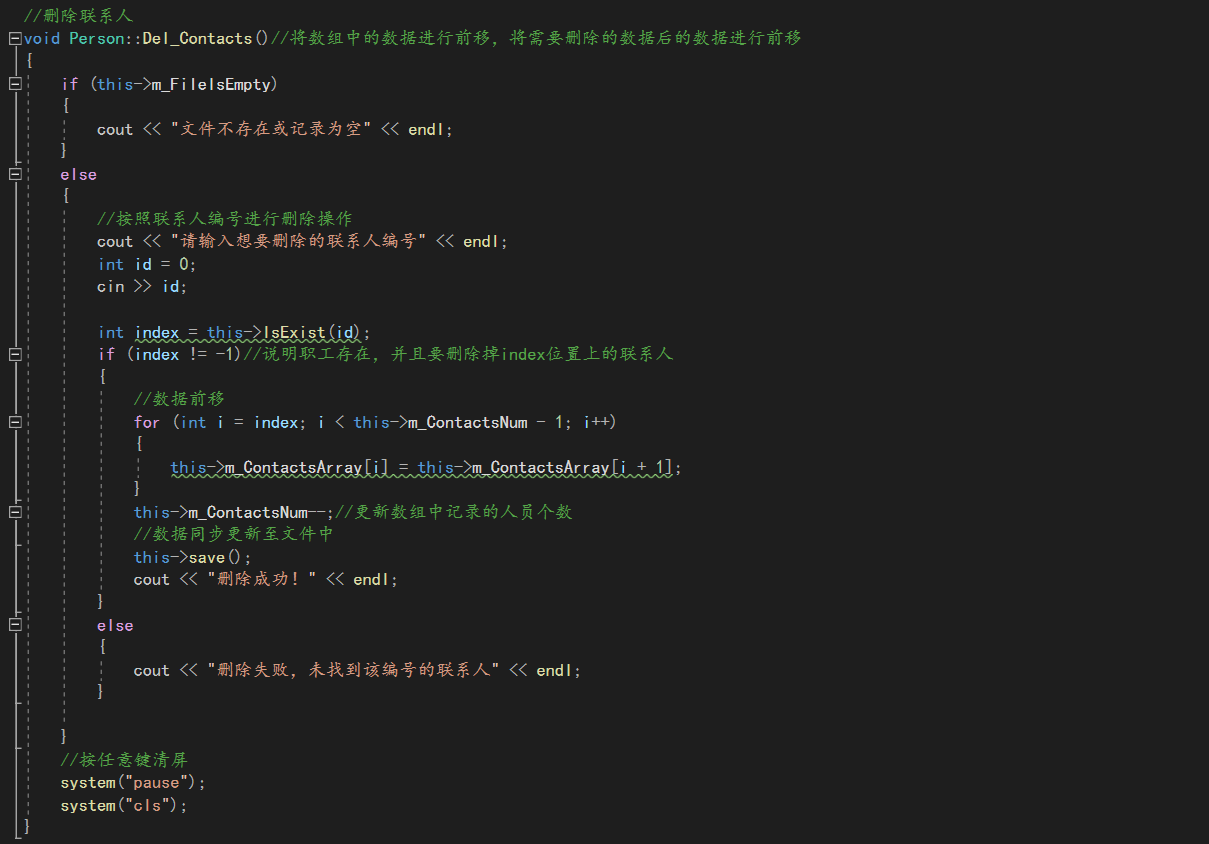
首先判断存储联系人信息的文件是否存在，若存在，再判断联系人的人数是否为零，如果不为零，则通过函数进行访问输出联系人的信息，否则无法对于联系人的信息进行显示。如图所示：

图十三

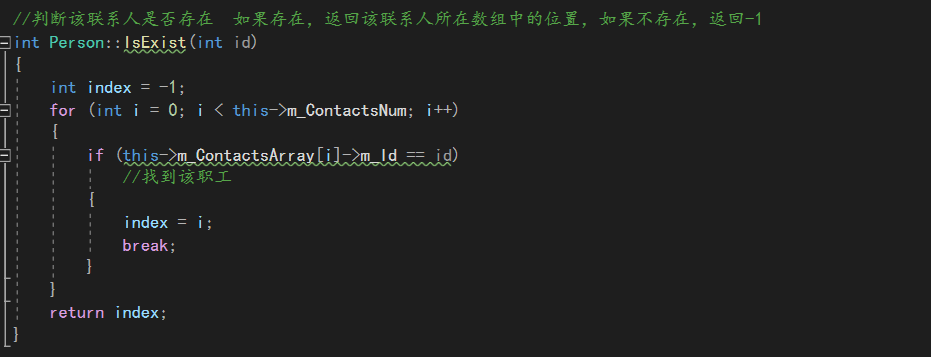
1. 删除联系人操作。

首先判断所需要删除的联系人是否存在，若存在，则进行删除操作，反之输出未查找到该联系人。

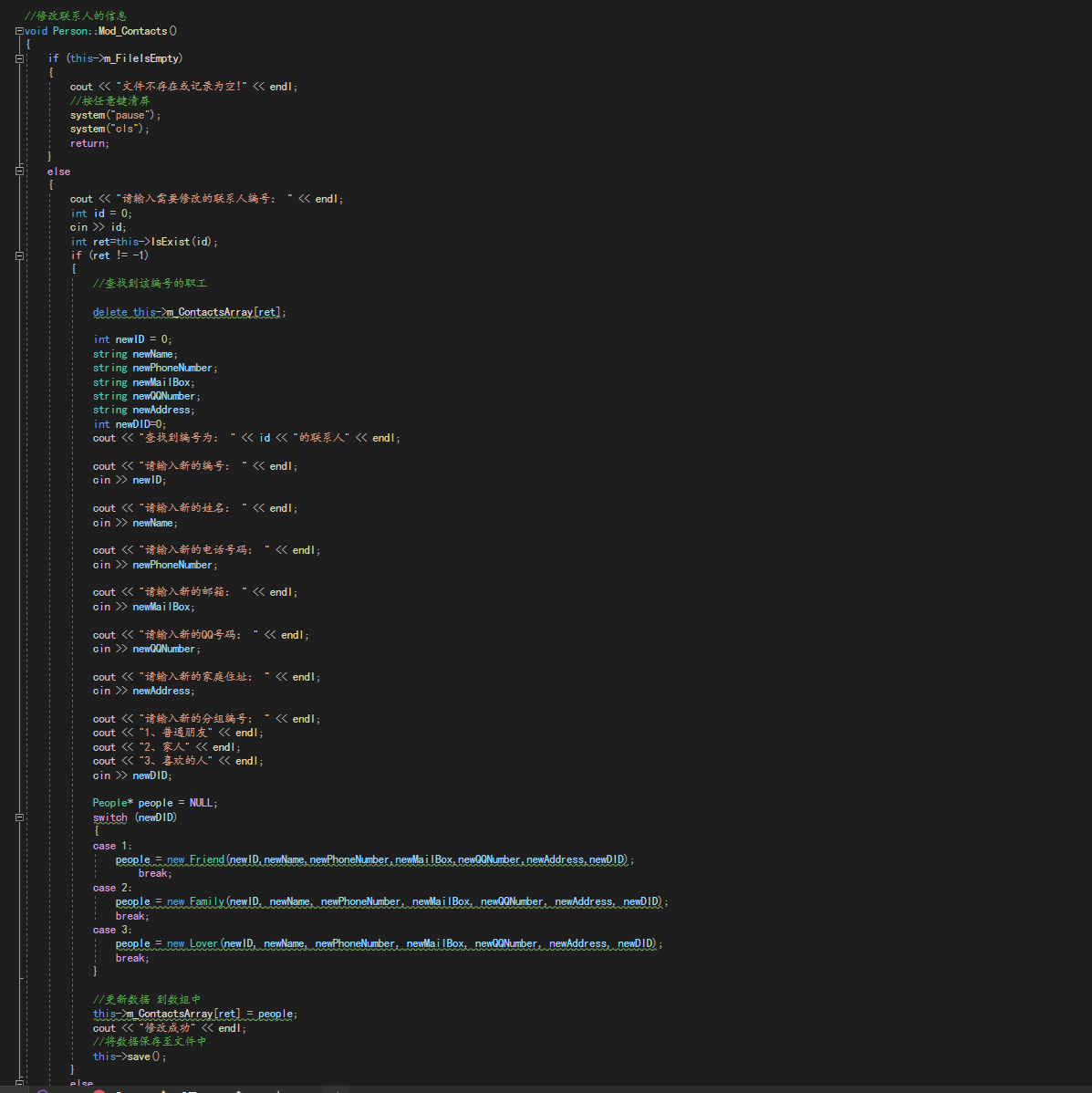
对于删除数据操作，从删除的位置开始对于每个数组中的数据进行前移操作将所需要删除的联系人信息进行删除。操作如图所示：

图十四

1. 修改联系人信息。

首先依然先对文件是否存在进行判断，之后通过【3】中写的函数判断所需要修改的联系人是否存在，若存在则返回职工在数组中的位置，然后对于该位置上的联系人信息进行修改。

图十五



图十六

1. 查找联系人。

判断文件或该联系人是否存在，若该联系人存在则通过函数返回联系人所在位置并且将联系人的信息进行打印输出。如图所示：



图十七

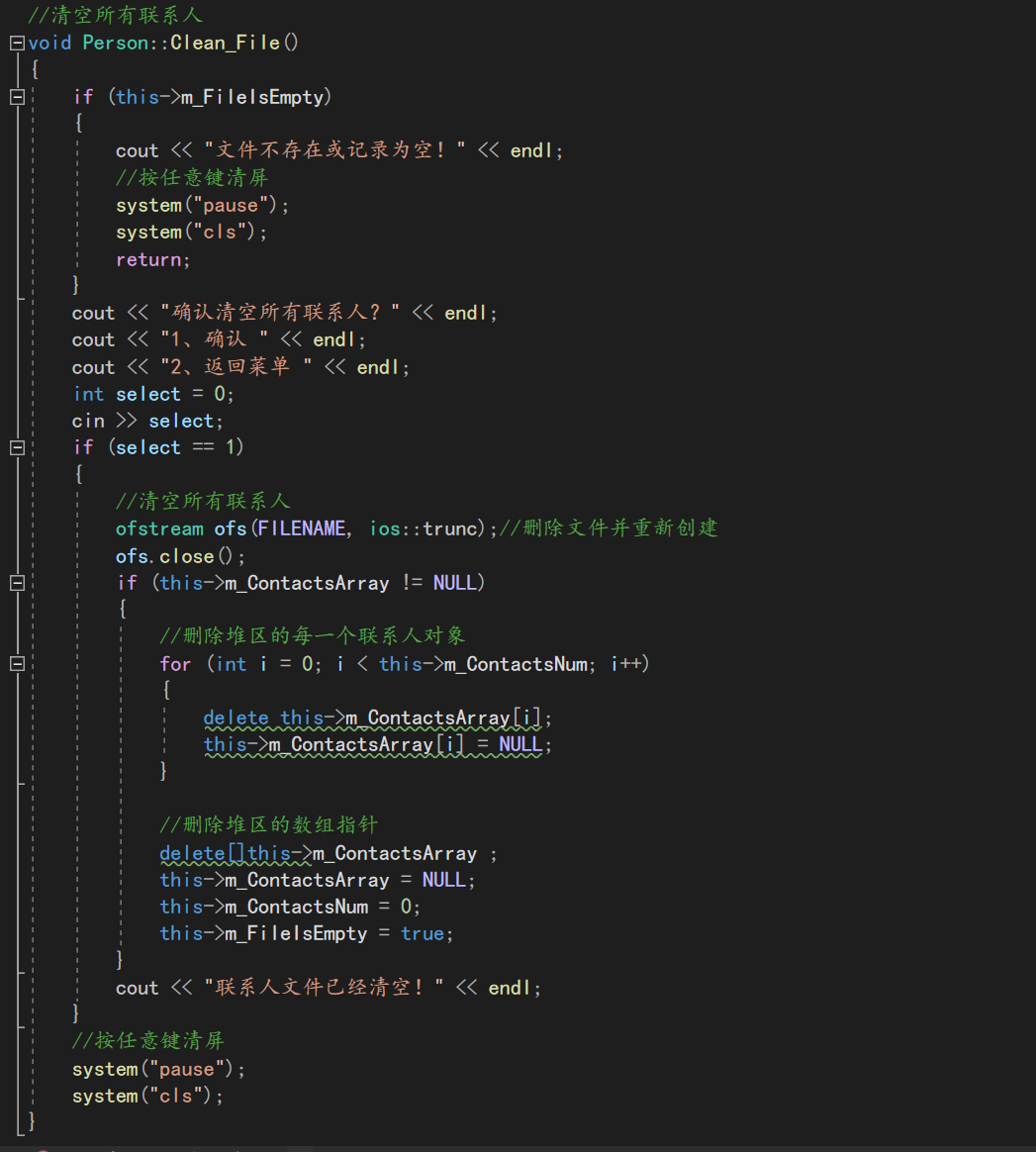
1. 按照序号对联系人进行排序。

通过选择排序法对联系人进行顺序或者逆序排序。如图所示：



图十八

1. 清空所有联系人。

使用删除文件并且重新创建的方式将所有联系人的信息进行清空操作。

图十九

People（父类）

此类代表所有联系人的虚类，里面包含了各种联系人所可能纯在的属性以及对于这些联系人属性

的输出以及显示的普通函数。

Person（父类）

在该类中实现对于联系人的显示修改删除等一系列操作，同时实例化出一个pe对象来进行操作

Lover（子类）

在Lover下的构造函数来对于所有信息初始化

采用多态的方式将联系人的信息进行打印输出

Friend（子类）

在Friend下的构造函数来对于所有信息初始化

采用多态的方式将联系人的信息进行打印输出

Family（子类）

在Family下的构造函数来对于所有信息初始化

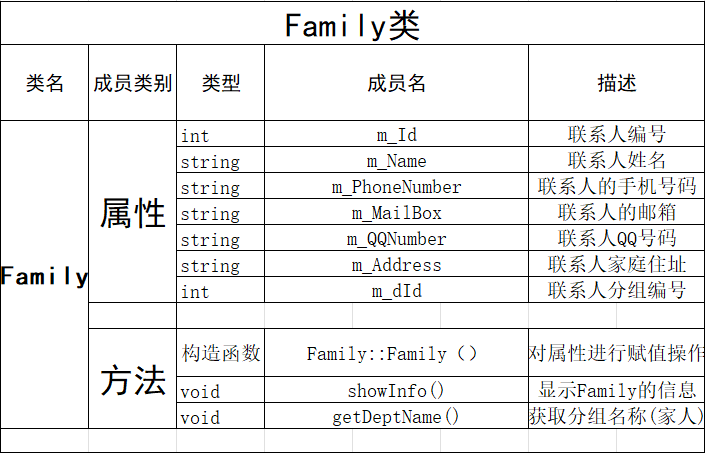
采用多态的方式将联系人的信息进行打印输出

**属性和方法定义一：**

**表一：联系人抽象类**

****

**表二：家人类**

****

**表三：朋友类**

****

**表四：亲密关系类**

****

**属性和方法定义二：**

**Person类：(对所有联系人系统的操作进行实现)**

****

## 3 用户手册

程序运行时，首先展示出菜单界面供用户进行选择。

选择0选项，退出该系统。

选择1选项，增加联系人的信息，其中包括联系人的编号，姓名，电话号码等重要信息。

选择2选项，显示目前所有已经添加过的联系人的信息，将所有联系人的信息进行展示和说明。

选择3选项，删除联系人的所有信息，在进行该项操作时，若通讯录中未记录该联系人的信息，那么将输出删除失败。

选择4选项，可以对于已经存储在联系人管理系统中的联系人信息进行修改。

选择5选项，可以对于存储在联系人管理系统中的联系人信息进行精准查找。

选择6选项，可以按照联系人的序号对所有联系人进行排序。

选择7选项，将联系人管理系统中所有联系人的信息进行删除操作，此操作是不可逆的。

## 4 调试及测试

对程序进行运行之后将每一个选项都进行适用，同时注意部分选项在进行使用时的注意事项。

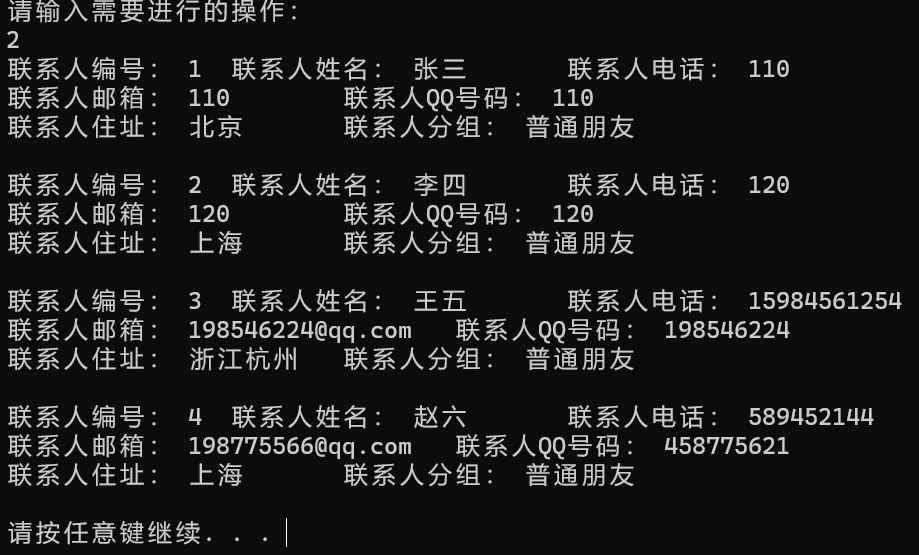
0:退出此程序。



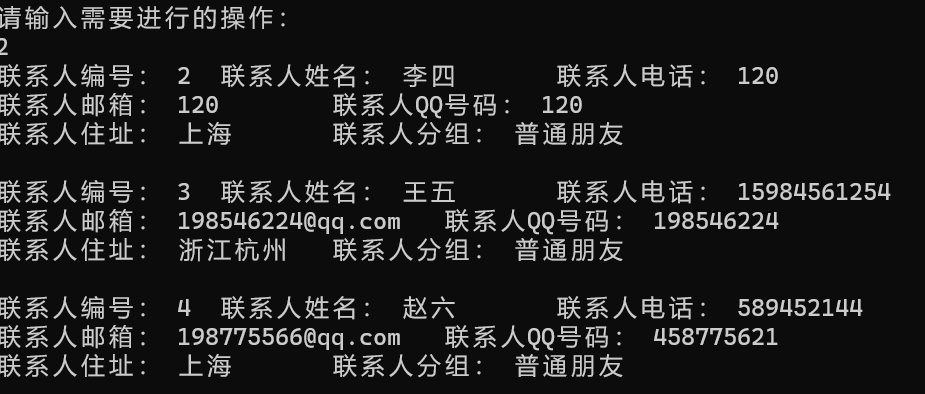
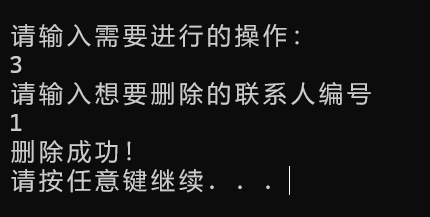
1：增加联系人的信息



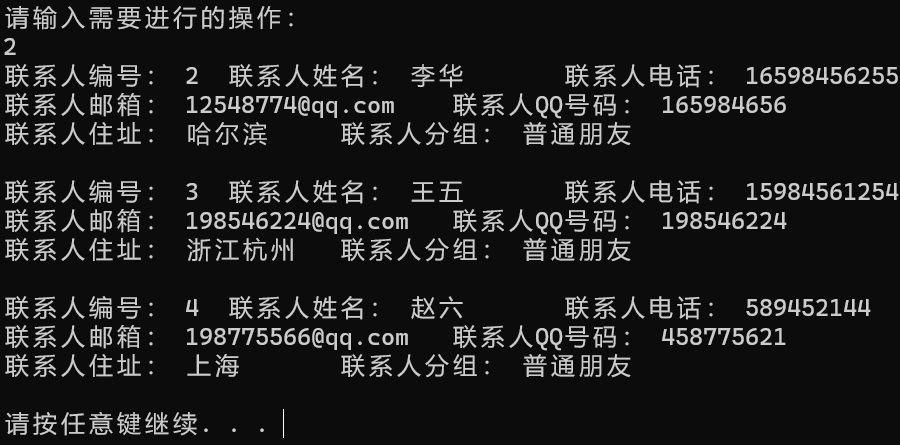
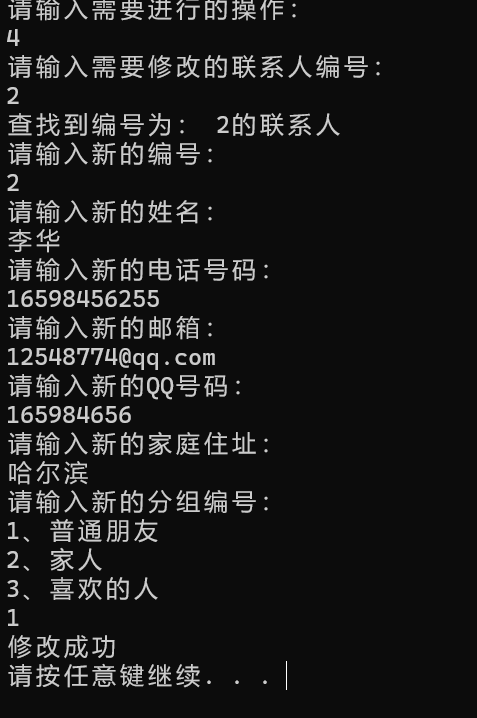
2：显示联系人信息



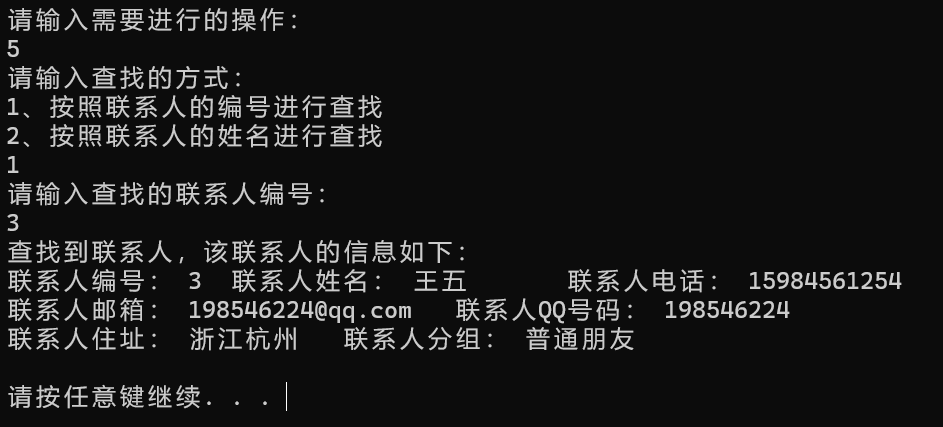
3：删除联系人操作



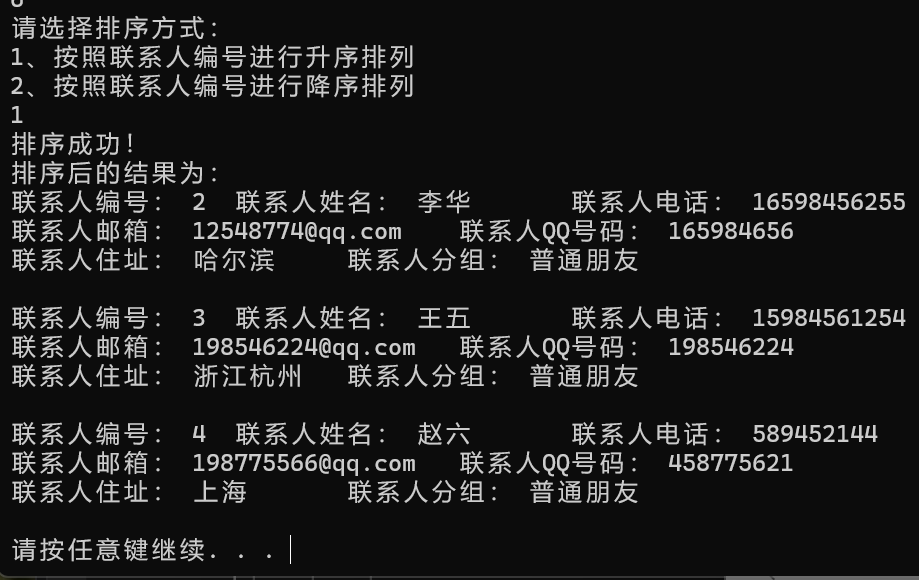
4:修改联系人信息



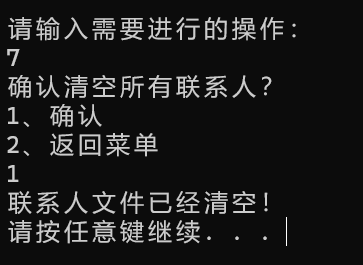
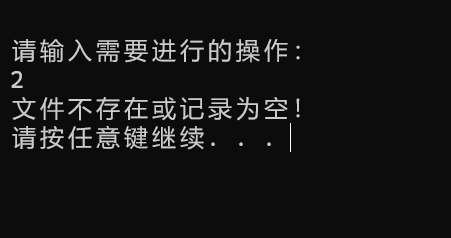
5：查找联系人



6：按照序号对联系人进行排序操作



7：清空所有联系人



## 5 运行实例

0：退出系统

1：请输入需要添加的联系人姓名：

请输入需要添加的联系人编号：

2：将所有联系人的信息进行显示说明

3：请输入联系人的姓名：

删除成功/失败

4：请输入需要修改的联系人编号：

修改成功/失败

5：请输入需要查找的联系人编号/姓名：

显示该联系人的所有信息

6：按照序号对所有联系人信息进行打印：

7：是否确认清空所有联系人

## 6 进一步改进

（1）目前程序中，在添加联系人的时候可能出现序号重复的现象，因此需要对于添加的序号进行一个判断，看看在管理系统中是否已经有该序号成员。

（2）若存储所有联系人信息的文件不存在，则需要重新创建一个新文件然后再将所有的信息添加上去。

## 7 心得体会

（可描述课程设计过程中出现的主要问题、原因及解决方法，课程设计的主要收获等）

在这次的课程设计中，更加深刻的理解了面对对象编程的方便性和实用性，通过面对对象，创建不同的类，在不同的类中封装不同的函数，对实现各个操作。

## 8 对课程设计的建议

希望增加更多有趣味的课程设计题目。

## 9 附录1 参考文献

CSDN面向对象编程方法。

黑马程序员c++联系人管理系统。

C++语言程序设计基础【清华大学第五版】

## 10 附录2⎯⎯源程序

主程序：Person.cpp文件

#include"Person.h"

using namespace std;

Person::Person()

{

//文件不存在

ifstream ifs;

ifs .open(FILENAME, ios::in);//读取文件

if (!ifs.is\_open())

{

//cout << "联系人文件不存在!" << endl;

//初始化联系人数组的人数为0

this->m\_ContactsNum = 0;

//初始化数组的指针指向空

this->m\_ContactsArray = NULL;

//初始化文件是否为空

this->m\_FileIsEmpty = true;

ifs.close();

return;

}

//文件存在但是文件内容为空

char ch;

ifs >> ch;

if (ifs.eof())//读取文件的第一个字符，如果是eof,则证明文件为空

{

//代表文件为空

//cout << "未记录任何联系人！" << endl;

//初始化联系人数组的人数为0

this->m\_ContactsNum = 0;

//初始化数组的指针指向空

this->m\_ContactsArray = NULL;

//初始化文件是否为空

this->m\_FileIsEmpty = true;

ifs.close();

return;

}

//当文件存在并且记录了数据

int num = this->get\_ContactsNum();

//cout << "职工的人数为： " << num << endl;

this->m\_ContactsNum = num;

this->m\_FileIsEmpty = false;

//开辟空间保证后续可以将原来的进行保存

this->m\_ContactsArray = new People \* [this->m\_ContactsNum];

//将文件中的数据，存放在数组中

this->init\_Contacts();

/\*for (int i = 0; i < this->m\_ContactsNum; i++)

{

cout << "联系人编号： " << this->m\_ContactsArray[i]->m\_Id << endl;

cout << "联系人姓名： " << this->m\_ContactsArray[i]->m\_Name << endl;

cout << "联系人电话： " << this->m\_ContactsArray[i]->m\_PhoneNumber << endl;

cout << "联系人邮箱： " << this->m\_ContactsArray[i]->m\_MailBox << endl;

cout << "联系人QQ号码： " << this->m\_ContactsArray[i]->m\_QQNumber << endl;

cout << "联系人地址： " << this->m\_ContactsArray[i]->m\_Address << endl;

cout << "联系人分组编号： " << this->m\_ContactsArray[i]->m\_dId << endl;

}\*/

}

//统计添加联系人之前的联系人人数

int Person::get\_ContactsNum()

{

ifstream ifs;

ifs.open(FILENAME, ios::in);//打开文件 读取文件

//联系人编号

int id;

//联系人姓名

string name;

//联系人的手机号码

string phonenumber;

//联系人的邮箱

string mailbox;

//联系人QQ号码

string qqnumber;

//联系人家庭住址

string address;

//联系人分组编号

int dId;

int num = 0;

while (ifs >> id && ifs >> name && ifs >> phonenumber && ifs >> mailbox && ifs >> qqnumber && ifs >> address && ifs >> dId)

{

//统计人数变量

num++;

}

return num;

}

//展示菜单

void Person::Show\_Menu()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\* 欢迎使用联系人管理系统! \*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0、退出该管理系统 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1、增加联系人信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2、显示联系人信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3、删除联系人信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4、修改联系人信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5、查找联系人信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6、按照其序号排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 7、清空所有联系人 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

}

//退出系统

void Person::ExitSystem()

{

cout << "欢迎下次使用" << endl;

system("pause");

exit(0);//退出程序

}

//判断职工编号是否存在函数

int Person::idIsExit(int id)

{

for (int i = 0; i < this->m\_ContactsNum; i++)

{

if (id == this->m\_ContactsArray[i]->m\_Id)

{

return 1;

}

}

return 0;

}

//添加联系人函数

void Person::Add\_Contacts()

{

cout << "请输入需要添加的联系人人数： " << endl;

int addNum = 0;//保存用户的输入数量

cin >> addNum;

int exist = 0;//判断新加入的职工编号是否存在

if (addNum > 0)

{

//添加

//计算添加新的空间大小

int newSize = this->m\_ContactsNum + addNum;//新空间的人数=原来记录的人数+新增加的人数

//开辟新空间

People\*\* newSpace = new People \* [newSize];

//将原来空间下的数据，拷贝到新空间下

if (this->m\_ContactsArray != NULL)

{

for (int i = 0; i < this->m\_ContactsNum; i++)

{

newSpace[i] = this->m\_ContactsArray[i];

}

}

//添加新联系人

for (int i = 0; i < addNum; i++)

{

//联系人编号

int id;

//联系人姓名

string name;

//联系人的手机号码

string phonenumber;

//联系人的邮箱

string mailbox;

//联系人QQ号码

string qqnumber;

//联系人家庭住址

string address;

//联系人分组选项

int dSelect;

back:

cout << "请输入第" << i + 1 << "个新联系人的编号： " << endl;

cin >> id;

exist = idIsExit(id);

if (exist == 1)

{

cout << "该编号已被使用，请更换新的联系人编号" << endl;

goto back;

}

cout << "请输入第" << i + 1 << "个新联系人的姓名： " << endl;

cin >> name;

cout << "请输入第" << i + 1 << "个新联系人的手机号码： " << endl;

cin >> phonenumber;

cout << "请输入第" << i + 1 << "个新联系人的邮箱： " << endl;

cin >> mailbox;

cout << "请输入第" << i + 1 << "个新联系人的QQ号码： " << endl;

cin >> qqnumber;

cout << "请输入第" << i + 1 << "个新联系人的家庭住址： " << endl;

cin >> address;

cout << "请选择该联系人的分组：" << endl;

cout << "1、普通朋友" << endl;

cout << "2、家人" << endl;

cout << "3、爱人" << endl;

cin >> dSelect;

People\* people = NULL;

switch (dSelect)

{

case 1:

people = new Friend( id, name, phonenumber, mailbox, qqnumber, address, 1);

break;

case 2:

people= new Family(id, name, phonenumber, mailbox, qqnumber, address, 2);

break;

case 3:

people= new Lover(id, name, phonenumber, mailbox, qqnumber, address, 3);

break;

default :

break;

}

//将创建的联系人，保存至数组中

newSpace[this->m\_ContactsNum + i] = people;

}

//释放原有空间

delete[]this->m\_ContactsArray;

//更改新空间指向

this->m\_ContactsArray = newSpace;

//更新新的联系人人数

this->m\_ContactsNum = newSize;

//保存数据内容至文件中

this->save();

//提示添加成功

cout << "成功添加了" << addNum << "名新联系人!" << endl;

}

else

{

cout << "输入数据有误！" << endl;

}

//按任意键清屏

system("pause");

system("cls");

}

//将修改完成的数据保存在文件中

void Person::save()

{

ofstream ofs;

ofs.open(FILENAME, ios::out);//用输出的方式来打开文件 写文件

//将每个联系人的数据写入到文件中

for (int i = 0; i < this->m\_ContactsNum; i++)

{

ofs << this->m\_ContactsArray[i]->m\_Id << " "

<< this->m\_ContactsArray[i]->m\_Name << " "

<< this->m\_ContactsArray[i]->m\_PhoneNumber << " "

<< this->m\_ContactsArray[i]->m\_MailBox << " "

<< this->m\_ContactsArray[i]->m\_QQNumber << " "

<< this->m\_ContactsArray[i]->m\_Address << " "

<< this->m\_ContactsArray[i]->m\_dId << endl;

}

//关闭文件

ofs.close();

}

//初始化联系人

void Person::init\_Contacts()

{

ifstream ifs;

ifs.open(FILENAME, ios::in);

//联系人编号

int id;

//联系人姓名

string name;

//联系人的手机号码

string phonenumber;

//联系人的邮箱

string mailbox;

//联系人QQ号码

string qqnumber;

//联系人家庭住址

string address;

//联系人分组编号

int dId;

int index = 0;//表示放置在联系人数组中的位置

People\* people = NULL;

while (ifs >> id && ifs >> name && ifs >> phonenumber && ifs >> mailbox && ifs >> qqnumber && ifs >> address && ifs >> dId)

{

if (dId == 1)//普通朋友

{

people = new Friend(id, name, phonenumber, mailbox, qqnumber, address, dId);

}

else if (dId == 2)//家人

{

people =new Family(id, name, phonenumber, mailbox, qqnumber, address, dId);

}

else if (dId == 3)//情侣

{

people= new Lover(id, name, phonenumber, mailbox, qqnumber, address, dId);

}

this->m\_ContactsArray[index]=people;//将所有原来的数据进行保存否则未记录任何数据

this->m\_FileIsEmpty = false;

index++;

}

ifs.close();//关闭文件

}

//显示所有联系人

void Person::Show\_Contacts()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

for (int i = 0; i < m\_ContactsNum; i++)

{

//利用多态调用程序接口

this->m\_ContactsArray[i]->showInfo();

}

}

system("pause");

system("cls");

}

//删除联系人

void Person::Del\_Contacts()//将数组中的数据进行前移，将需要删除的数据后的数据进行前移

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空" << endl;

}

else

{

//按照联系人编号进行删除操作

cout << "请输入想要删除的联系人编号" << endl;

int id = 0;

cin >> id;

int index = this->IsExist(id);

if (index != -1)//说明职工存在，并且要删除掉index位置上的联系人

{

//数据前移

for (int i = index; i < this->m\_ContactsNum - 1; i++)

{

this->m\_ContactsArray[i] = this->m\_ContactsArray[i + 1];

}

this->m\_ContactsNum--;//更新数组中记录的人员个数

//数据同步更新至文件中

this->save();

cout << "删除成功！" << endl;

}

else

{

cout << "删除失败，未找到该编号的联系人" << endl;

}

}

//按任意键清屏

system("pause");

system("cls");

}

//判断该联系人是否存在 如果存在，返回该联系人所在数组中的位置，如果不存在，返回-1

int Person::IsExist(int id)

{

int index = -1;

for (int i = 0; i < this->m\_ContactsNum; i++)

{

if (this->m\_ContactsArray[i]->m\_Id == id)

//找到该职工

{

index = i;

break;

}

}

return index;

}

//修改联系人的信息

void Person::Mod\_Contacts()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空!" << endl;

//按任意键清屏

system("pause");

system("cls");

return;

}

else

{

cout << "请输入需要修改的联系人编号： " << endl;

int id = 0;

cin >> id;

int ret=this->IsExist(id);

if (ret != -1)

{

//查找到该编号的职工

delete this->m\_ContactsArray[ret];

int newID = 0;

string newName;

string newPhoneNumber;

string newMailBox;

string newQQNumber;

string newAddress;

int newDID=0;

cout << "查找到编号为： " << id << "的联系人" << endl;

cout << "请输入新的编号： " << endl;

cin >> newID;

cout << "请输入新的姓名： " << endl;

cin >> newName;

cout << "请输入新的电话号码： " << endl;

cin >> newPhoneNumber;

cout << "请输入新的邮箱： " << endl;

cin >> newMailBox;

cout << "请输入新的QQ号码： " << endl;

cin >> newQQNumber;

cout << "请输入新的家庭住址： " << endl;

cin >> newAddress;

cout << "请输入新的分组编号： " << endl;

cout << "1、普通朋友" << endl;

cout << "2、家人" << endl;

cout << "3、喜欢的人" << endl;

cin >> newDID;

People\* people = NULL;

switch (newDID)

{

case 1:

people = new Friend(newID,newName,newPhoneNumber,newMailBox,newQQNumber,newAddress,newDID);

break;

case 2:

people = new Family(newID, newName, newPhoneNumber, newMailBox, newQQNumber, newAddress, newDID);

break;

case 3:

people = new Lover(newID, newName, newPhoneNumber, newMailBox, newQQNumber, newAddress, newDID);

break;

}

//更新数据 到数组中

this->m\_ContactsArray[ret] = people;

cout << "修改成功" << endl;

//将数据保存至文件中

this->save();

}

else

{

cout << "查无此人，修改失败！" << endl;

}

//按任意键清屏

system("pause");

system("cls");

}

}

//查找联系人函数

void Person::Find\_Contacts()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

back2:

cout << "请输入查找的方式： " << endl;

cout << "1、按照联系人的编号进行查找 " << endl;

cout << "2、按照联系人的姓名进行查找 " << endl;

int select = 0;

cin >> select;

switch (select)

{

case 1://按照编号方式进行查找

{

int id = 0;

cout << "请输入查找的联系人编号： " << endl;

cin >> id;

int ret=IsExist(id);

if (ret != -1)

{

//查找到此联系人

cout << "查找到联系人，该联系人的信息如下： " << endl;

this->m\_ContactsArray[ret]->showInfo();

}

else

{

cout << "查找失败，查无此人!" << endl;

}

break;

}

case 2://按照姓名方式进行查找

{

string name;

cout << "请输入需要查找联系人的姓名： " << endl;

cin >> name;

//判断是否查到的标志

bool find = false;//默认未找到该联系人

for (int i = 0; i < this->m\_ContactsNum; i++)

{

if (name == this->m\_ContactsArray[i]->m\_Name)

{

cout << "查找到联系人，该联系人的信息如下： " << endl;

//调用显示信息的接口

this->m\_ContactsArray[i]->showInfo();

find = true;

}

}

if (find == false)

{

cout << "查找失败，查无此人！" << endl;

}

break;

}

default:

{

cout << "输入错误，请重新输入！" << endl;

goto back2;

break;

}

}

}

//按任意键清屏操作

system("pause");

system("cls");

}

//排序联系人

void Person::Sort\_Contacts()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

//按任意键清屏

system("pause");

system("cls");

return;

}

else

{

back2:

cout << "请选择排序方式： " << endl;

cout << "1、按照联系人编号进行升序排列" << endl;

cout << "2、按照联系人编号进行降序排列" << endl;

int select = 0;

cin >> select;

if (select != 1 && select != 2)

{

cout << "输入错误请重新输入！" << endl;

goto back2;

}

for (int i = 0; i < this->m\_ContactsNum; i++)

{

int minOrMax = i;//声明最小值或者最大值

for (int j = i + 1; j < this->m\_ContactsNum; j++)

{

if (select == 1)//升序排列

{

if (this->m\_ContactsArray[minOrMax]->m\_Id > this->m\_ContactsArray[j]->m\_Id)

{

minOrMax = j;

}

}

else if (select == 2)//降序排列

{

if (this->m\_ContactsArray[minOrMax]->m\_Id < this->m\_ContactsArray[j]->m\_Id)

{

minOrMax = j;

}

}

}

//判断一开始认定的最大值或者最小值是否为计算所得的最大值或者最小值

//如果不是，交换数据

if (i != minOrMax)

{

People\* temp = this->m\_ContactsArray[i];

this->m\_ContactsArray[i] = this->m\_ContactsArray[minOrMax];

this->m\_ContactsArray[minOrMax] = temp;

}

}

}

cout << "排序成功！" << endl;

cout << "排序后的结果为： " << endl;

this->save();//将排序后的结果保存至文件中

this->Show\_Contacts();//显示所有的联系人

}

//清空所有联系人

void Person::Clean\_File()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

//按任意键清屏

system("pause");

system("cls");

return;

}

cout << "确认清空所有联系人？" << endl;

cout << "1、确认 " << endl;

cout << "2、返回菜单 " << endl;

int select = 0;

cin >> select;

if (select == 1)

{

//清空所有联系人

ofstream ofs(FILENAME, ios::trunc);//删除文件并重新创建

ofs.close();

if (this->m\_ContactsArray != NULL)

{

//删除堆区的每一个联系人对象

for (int i = 0; i < this->m\_ContactsNum; i++)

{

delete this->m\_ContactsArray[i];

this->m\_ContactsArray[i] = NULL;

}

//删除堆区的数组指针

delete[]this->m\_ContactsArray ;

this->m\_ContactsArray = NULL;

this->m\_ContactsNum = 0;

this->m\_FileIsEmpty = true;

}

cout << "联系人文件已经清空！" << endl;

}

//按任意键清屏

system("pause");

system("cls");

}

Person::~Person()

{

//堆区内容手动开辟，手动释放

if (this->m\_ContactsArray != NULL)

{

for (int i = 0; i < this->m\_ContactsNum; i++)

{

if (this->m\_ContactsArray[i] != NULL)

{

delete this->m\_ContactsArray[i];

}

}

delete[]this->m\_ContactsArray;

this->m\_ContactsArray = NULL;

}

}

Person.h的头文件：

#pragma once//防止头文件重复包含

#include<iostream>//包含输入输出流头文件

using namespace std;//使用标准命名空间

#include"People.h"//联系人的抽象类，其中包括三种分类

#include"Friend.h"

#include"Family.h"

#include"Lover.h"

#include<fstream>

#define FILENAME "contacts.txt"

class Person

{

public:

//构造函数

Person();

//展示菜单

void Show\_Menu();

//退出系统

void ExitSystem();

//记录联系人的人数

int m\_ContactsNum;

//联系人数组指针

People\*\* m\_ContactsArray;

//添加联系人函数

void Add\_Contacts();

//判断职工编号是否存在函数

int idIsExit(int id);

//保存文件

void save();

//标志文件是否为空

bool m\_FileIsEmpty=false;//初始化文件不为空

//统计添加联系人之前的联系人人数

int get\_ContactsNum();

//初始化联系人

void init\_Contacts();

//显示所有联系人

void Show\_Contacts();

//删除联系人

void Del\_Contacts();

//判断该联系人是否存在 如果存在，返回该联系人所在数组中的位置，如果不存在，返回-1

int IsExist(int id);

//修改联系人的信息

void Mod\_Contacts();

//查找联系人

void Find\_Contacts();

//排序联系人

void Sort\_Contacts();

//清空所有联系人

void Clean\_File();

//析构函数

~Person();

};

People.h的头文件：

#pragma once

#include<iostream>

using namespace std;

#include<string>

//联系人抽象类(父类)

class People

{

public:

//显示联系人的所有信息

virtual void showInfo() = 0;

//获取分组名称

virtual string getDeptName() = 0;

//联系人编号

int m\_Id;

//联系人姓名

string m\_Name;

//联系人的手机号码

string m\_PhoneNumber;

//联系人的邮箱

string m\_MailBox;

//联系人QQ号码

string m\_QQNumber;

//联系人家庭住址

string m\_Address;

//联系人分组编号

int m\_dId;

};

Family.h的头文件：

#pragma once

#include<iostream>

using namespace std;

#include"People.h"

//家人类

class Family :public People

{

public:

//构造函数

Family(int id, string name, string phonenumber, string mailbox, string qqnumber, string address, int dId);

//显示联系人的所有信息

virtual void showInfo() ;

//获取分组名称

virtual string getDeptName() ;

};

Friend.h的头文件：

#pragma once

#include<iostream>

using namespace std;

#include"People.h"

//朋友类

class Friend :public People

{

public:

//构造函数

Friend(int id,string name,string phongnumber,string mailbox,string qqnumber,string address,int dId);

//显示联系人的所有信息

void showInfo();

//获取分组名称

string getDeptName();

};

Lover.h的头文件：

#pragma once

#include<iostream>

using namespace std;

#include"People.h"

//情侣类

class Lover :public People

{

public:

//构造函数

Lover(int id, string name, string phonenumber, string mailbox, string qqnumber, string address, int dId);

//显示联系人的所有信息

virtual void showInfo();

//获取分组名称

virtual string getDeptName();

};

联系人管理系统的cpp文件：（实例化Person对象对函数进行调用）

#include<iostream>

using namespace std;

#include"Person.h"

#include"People.h"

#include"Friend.h"

#include"Family.h"

#include"Lover.h"

int main()

{

//实例化对象

Person pe;

int choice = 0;//用来存储用户的选项

while (true)

{

//调用展示菜单成员函数

pe.Show\_Menu();

cout << "请输入需要进行的操作：" << endl;

cin >> choice;

switch (choice)

{

case 0://退出系统

pe.ExitSystem();

break;

case 1://增加联系人

pe.Add\_Contacts();

break;

case 2://显示联系人

pe.Show\_Contacts();

break;

case 3://删除联系人

pe.Del\_Contacts();

break;

case 4://修改联系人

pe.Mod\_Contacts();

break;

case 5://查找联系人

pe.Find\_Contacts();

break;

case 6://按照序号排序联系人

pe.Sort\_Contacts();

break;

case 7://清空所有联系人

pe.Clean\_File();

break;

default :

system("cls");//清屏操作

break;

}

}

system("pause");

return 0;

}