目录

[一、设计要求 1](#_Toc517980360)

[二、总体设计 2](#_Toc517980361)

[三、界面设计与各功能模块实现 3](#_Toc517980362)

[四、使用说明 9](#_Toc517980363)

[五、附件 10](#_Toc517980364)

[1.头文件 10](#_Toc517980365)

[1）手机联系人类 //phonecontact.h 10](#_Toc517980366)

[2）手机卡联系人类 //cardcontact.h 10](#_Toc517980367)

[3）通讯簿抽象类 //maillist.h 11](#_Toc517980368)

[4）手机通讯簿类 //phonemaillist.h 11](#_Toc517980369)

[5）手机卡通讯簿类 //cardmaillist.h 12](#_Toc517980370)

[6）用户类 //user.h 13](#_Toc517980371)

[7）菜单类 //uimenu.h 13](#_Toc517980372)

[2. 除主函数外的源文件 14](#_Toc517980373)

[1）手机联系人类的实现 //PhoneContact.cpp 14](#_Toc517980374)

[2）手机卡联系人类的实现 //CardContact.cpp 16](#_Toc517980375)

[3）手机通讯簿类的实现 //PhoneMaillist.cpp 18](#_Toc517980376)

[4）手机卡通讯簿类的实现 //CardMaillist.cpp 24](#_Toc517980377)

[5）用户类的实现 //User.cpp 31](#_Toc517980378)

[6）菜单类的实现 //Uimenu.cpp 38](#_Toc517980379)

[3. 主函数源文件 //Main.cpp 40](#_Toc517980380)

# 一、设计要求

1. 创建文本文件，记录联系人的信息（需要创建两个文本文件，分别存储手机和手机卡上的存储的联系人的信息）。
2. 以菜单方式工作(字符界面即可)
3. 存储在手机上的联系人的信息包括：姓名和电话号码；存储在手机卡上的联系人的信息包括姓名，籍贯，电话号码，QQ号等信息
4. 管理系统的功能包括：
   1. 新建联系人：添加新的联系人（添加时确定是添加到手机上还是手机卡中）
   2. 删除：删除一个联系人（输入电话号码，删除该联系人。说明，如果两个存储位置上都存在该联系人的话，需要在两个存储位置上都要进行删除操作）
   3. 修改：修改某个联系人的信息（输入电话号码，查询到联系人之后进行信息的修改。说明，如果两个存储位置上都存在该联系人的话，需要在两个存储位置上都要进行修改操作）
   4. 查询：根据名字查询联系人信息（查询结果不唯一）
   5. 浏览：显示所有联系人的信息
   6. 将联系人的信息从手机转存到手机卡上（同时要避免重复数据的存在。并且在转存是要检查容量是否受限。下同。）
   7. 将联系人的信息从手机卡上准存到手机上（同时要避免重复数据的存在）
5. 要支持继承、多态、重载(运算符重载、函数重载)等面向对象的基本特点

# 二、总体设计

1.本设计由14个文件组成，分别为7个头文件及相应的cpp文件。

2.为了便于数据存储与使用，联系人姓名、号码、籍贯、QQ均使用string类进行操作；

3.为了易于进行联系人新建、删除、修改、查询等操作，故用于保存数据的底层数据结构是向量数组，使用C++ STL库提供的vector来实现这个向量数组的结构；

4.程序的核心为手机通讯簿类与手机卡通讯簿类，实质是具有手机联系人对象与具有手机卡联系人对象的两个数据库，其成员函数用于实现通讯录系统的增删改查；

5.程序的操作由菜单类与用户类实现，菜单类提供操作界面，用户类提供操作；用户类中两个对象分别为手机通讯簿类与手机卡通讯簿类，通过在用户类对这两个类的动态联编，从而实现在两个文本文件中的操作。

# 三、界面设计与各功能模块实现

**建立两个文本文件：**记录存储在手机上联系人的基本信息（文本文件：手机.txt）和存储在手机卡上的联系人的信息（文本文件：手机卡.txt）。

**建立以下几个类：**

**1.手机联系人类（PhoneContact）：表示一个手机联系人**

数据成员：

姓名：string name;

电话号码：string tel\_num;

成员函数：

并带默认值参数的构造函数：PhoneContact(string rname = "新联系人", string rtel\_num = "无号码");

修改数据成员值的函数：void modify();

重载>>运算符，完成对象的输入操作：friend istream &operator >>(istream &input, PhoneContact &obj);

重载<<运算符，完成对象的输出操作：friend ostream &operator <<(ostream &output, PhoneContact &obj);

获取姓名的函数：get\_name();

获取号码的函数：string get\_tel\_num();

为数据成员赋值的函数：void set(string rname, string rtel\_num);

**2.手机卡联系人（CardContact）：继承于手机联系人类，表示一个手机卡联系人**

新增数据成员：

籍贯：string homeland;

QQ号：string QQ;

成员函数：

并带默认值参数的构造函数：CardContact(string rname = "新联系人", string rtel\_num = "无号码", string rhomeland = "无地址", string rQQ = "无QQ");

修改数据成员值的函数：void modify();

重载>>运算符，完成对象的输入操作：friend istream & operator >>(istream &input, CardContact &obj);

重载<<运算符，完成对象的输出操作：friend ostream & operator <<(ostream &output, CardContact &obj);

获取籍贯的函数：string get\_homeland();

获取QQ的函数：string get\_QQ();

为数据成员赋值的函数：void set(string rname, string rtel\_num,string rhomeland,string rQQ);

**3.通讯簿抽象类（Maillist）：为支持多态，将函数封装为纯虚函数**

**数据成员：无**

**成员函数：**

新建联系人的函数：virtual void add() = 0;

删除联系人的函数：virtual void Delete(string rtel\_num) = 0;

修改联系人的信息：virtual void change(string rtel\_num) = 0;

查询并显示某一联系人的信息的函数：virtual void inquiry(string rname) = 0;

显示所有联系信息的函数：virtual void display() = 0;

**4.手机通讯簿类（PhoneMaillist）：继承于通讯簿抽象类，用于记录手机中存储的所有联系人的信息**

数据成员：

手机联系人的数量：int phone\_contact\_num;

手机联系人对象数组：vector <PhoneContact> phone\_contact;

成员函数：

构造函数：PhoneMaillist();    //读取文本文件（手机.txt）中的数据，并根据文件内容创建联系人对象数组

析构函数：~PhoneMaillist();    //将对象数组中的内容写入到文本文件（手机.txt）中

新建联系人的函数：void add();

删除联系人的函数：void Delete(string rtel\_num);

修改联系人的信息：void change(string rtel\_num);

查询并显示某一联系人的信息的函数：void inquiry(string rname);

显示所有联系信息的函数：void display();

获取手机通讯录联系人数量的函数：int get\_phone\_contact\_num();

获取动态数组第i个联系人的姓名的函数： string get\_name(int i);    //用于查询联系人信息

获取动态数组第i个联系人的号码的函数：string get\_tel\_num(int i);    //用于查询联系人信息

将姓名、号码压入动态数组的函数：void copy(string rname, string rtel\_num);    //用于两个通讯簿之间数据复制、移动

清空手机通讯录的函数：void clear();    //用于两个通讯簿之间数据复制、移动

**5.手机卡通信簿类（CardMaillist）：继承于通讯簿抽象类，用于记录手机卡中存储的所有联系人的信息**

数据成员：

手机卡联系人的数量：int card\_contact\_num;

手机卡联系人对象数组：vector <CardContact> card\_contact;

成员函数：

构造函数：CardMaillist();    //读取文本文件（手机卡.txt）中的数据，并根据文件内容创建联系人对象数组

析构函数：~CardMaillist();    //将对象数组中的内容写入到文本文件（手机卡.txt）中

新建联系人的函数：void add();

删除联系人的函数：void Delete(string rtel\_num);

修改联系人的信息：void change(string rtel\_num);

查询并显示某一联系人的信息的函数：void inquiry(string rname);

显示所有联系信息的函数：void display();

获取手机卡通讯录联系人数量的函数：int get\_card\_contact\_num();

获取动态数组第i个联系人的姓名的函数： string get\_name(int i);    //用于查询联系人信息

获取动态数组第i个联系人的号码的函数：string get\_tel\_num(int i);    //用于查询联系人信息

将姓名、号码压入动态数组的函数：void copy(string rname, string rtel\_num);    //用于两个通讯簿之间数据复制、移动

清空手机卡通讯录的函数：void clear();    //用于两个通讯簿之间数据复制、移动

**6.用户类（User）：对通讯录进行管理，利用基类指针，调用相应的通讯簿对象，从而实现动态联编。**

数据成员：

手机通讯簿类的对象： phonemaillist;

手机卡通讯簿类的对象：cardmaillist;

菜单类的对象：uimenu;

成员函数：

新建联系人的函数：void add();

删除联系人的函数：void Delete();

修改联系人的信息：void change();

查询并显示某一联系人的信息的函数：void inquiry();

显示所有联系信息的函数：void display();

移动手机卡通讯录到手机通讯录：void move\_card\_to\_phone();

移动手机通讯录到手机卡通讯录：void move\_phone\_to\_card();

复制手机卡通讯录到手机通讯录：void copy\_card\_to\_phone();

复制手机通讯录到手机卡通讯录：void copy\_phone\_to\_card();

**7.界面菜单类（Uimenu）：用来给出操作提示**

数据成员：无

成员函数：

主菜单：void MainUimenu();

添加菜单：void AddUimenu();

删除菜单：void DeleteUimenu();

修改菜单：void ChangeUimenu();

修改手机联系人菜单：void ChangePhoneUimenu();

修改手机卡联系人菜单：void ChangeCardUimenu();

查询菜单：void InquiryUimenu();

显示菜单：void DisplayUimenu();

转存菜单：void TransferredMethod();

# 四、使用说明

1. 程序开始时，用户类及其对象的构造函数会自动打开文件进行数据读入，结束时按0会自动关闭文件进行数据写入并保存；

2.本程序大量使用转向语句，使得任一模块均具有返回上一级及容错操作，操作过程按照提示操作即可；

3.本程序存在缺陷，不支持动态存储，只支持静态存储，即所有操作完成后按0结束程序才会写入数据。

# 五、附件

## 1.头文件

### 1）手机联系人类 //phonecontact.h

/\*手机联系人类\*/

#ifndef PHONECONTACT\_H

#define PHONECONTACT\_H

#include<iostream>

#include<string>

using namespace std;

class PhoneContact {

protected:

string name;

string tel\_num;

public:

PhoneContact(string rname = "新联系人", string rtel\_num = "无号码");    //有参构造函数

void modify(); //修改数据成员的值

friend istream &operator >>(istream &input, PhoneContact &obj);       //重载>>运算符

friend ostream &operator <<(ostream &output, PhoneContact &obj);     //重载<<运算符

string get\_name(); //获取姓名

string get\_tel\_num(); //获取号码

void set(string rname, string rtel\_num);  //为数据成员赋值

};

#endif

### 2）手机卡联系人类 //cardcontact.h

/\*手机卡联系人类\*/

#ifndef CARDCONTACT\_H

#define CARDCONTACT\_H

#include"phonecontact.h"

class CardContact : public PhoneContact{

protected:

string homeland;

string QQ;

public:

CardContact(string rname = "新联系人", string rtel\_num = "无号码", string rhomeland = "无地址", string rQQ = "无QQ");    //有参构造函数

void modify();    //修改数据成员的值

friend istream & operator >>(istream &input, CardContact &obj);         //重载>>运算符

friend ostream & operator <<(ostream &output, CardContact &obj);        //重载<<运算符

string get\_homeland();  //获取籍贯

string get\_QQ(); //获取QQ

void set(string rname, string rtel\_num,string rhomeland,string rQQ);    //为数据成员赋值

};

#endif

### 3）通讯簿抽象类 //maillist.h

/\*通讯簿抽象类\*/

#ifndef MAILLIST\_H

#define MAILLIST\_H

#include<iostream>

#include<string>

using namespace std;

class Maillist {

public:

virtual void add() = 0; //新建联系人

virtual void Delete(string rtel\_num) = 0; //删除联系人

virtual void change(string rtel\_num) = 0; //修改联系人信息

virtual void inquiry(string rname) = 0; //查询联系人信息

virtual void display() = 0; //显示所有联系人信息

};

#endif

### 4）手机通讯簿类 //phonemaillist.h

/\*手机通讯簿类\*/

#ifndef PHONEMAILLIST\_H

#define PHONEMAILLIST\_H

#include"cardcontact.h"

#include"maillist.h"

#include<vector>

class PhoneMaillist:public Maillist {

protected:

int phone\_contact\_num; //手机联系人数量

vector <PhoneContact> phone\_contact; //手机联系人对象动态数组

public:

PhoneMaillist(); //构造函数，读取文本文件中的数据,并根据文件内容创建联系人对象数组

~PhoneMaillist(); //析构函数，将对象数组中的内容写入到文本文件中

void add(); //新建联系人

void Delete(string rtel\_num); //删除联系人

void change(string rtel\_num); //修改联系人信息

void inquiry(string rname); //查询联系人信息

void display(); //显示所有联系人信息

int get\_phone\_contact\_num(); //获取手机通讯录联系人数量

void copy(string rname, string rtel\_num); //将姓名、号码压入动态数组

string get\_name(int i); //获取动态数组第i个联系人的姓名

string get\_tel\_num(int i); //获取动态数组第i个联系人的号码

void clear(); //清空手机通讯录

};

#endif

### 5）手机卡通讯簿类 //cardmaillist.h

/\*手机卡通讯簿类\*/

#ifndef CARDMAILLIST\_H

#define CARDMAILLIST\_H

#include"cardcontact.h"

#include"maillist.h"

#include<vector>

class CardMaillist :public Maillist {

protected:

int card\_contact\_num; //手机卡联系人数量

vector <CardContact> card\_contact; //手机卡联系人对象动态数组

public:

CardMaillist(); //构造函数，读取文本文件中的数据,并根据文件内容创建联系人对象数组

~CardMaillist(); //析构函数，将对象数组中的内容写入到文本文件中

void add() ; //新建联系人

void Delete(string rtel\_num) ; //删除联系人

void change(string rtel\_num) ; //修改联系人信息

void inquiry(string rname) ; //查询联系人信息

void display() ; //显示所有联系人信息

int get\_card\_contact\_num(); //获取手机卡通讯录联系人数量

void copy(string rname, string rtel\_num); //将姓名、号码压入动态数组

string get\_name(int i); //获取动态数组第i个联系人的姓名

string get\_tel\_num(int i); //获取动态数组第i个联系人的号码

void clear(); //清空手机卡通讯录

};

#endif

### 6）用户类 //user.h

/\*用户类\*/

#ifndef USER\_H

#define USER\_H

#include"cardmaillist.h"

#include"phonemaillist.h"

#include"uimenu.h"

class User {

PhoneMaillist phonemaillist;

CardMaillist cardmaillist;

Uimenu uimenu;

public:

User(); //构造函数

~User(); //析构函数

void add(); //新建联系人

void Delete(); //删除联系人

void change(); //修改联系人信息

void inquiry(); //查询联系人信息

void display(); //显示所有联系人信息

void move\_card\_to\_phone(); //移动手机卡通讯录到手机通讯录

void move\_phone\_to\_card(); //移动手机通讯录到手机卡通讯录

void copy\_card\_to\_phone(); //复制手机卡通讯录到手机通讯录

void copy\_phone\_to\_card(); //复制手机通讯录到手机卡通讯录

};

#endif

### 7）菜单类 //uimenu.h

/\*菜单类\*/

#ifndef UIMENU\_H

#define UIMENU\_H

#include<iostream>

using namespace std;

class Uimenu {

public:

void MainUimenu(); //主菜单

void AddUimenu(); //添加菜单

void DeleteUimenu(); //删除菜单

void ChangeUimenu(); //修改菜单

void ChangePhoneUimenu(); //修改手机联系人菜单

void ChangeCardUimenu(); //修改手机卡联系人菜单

void InquiryUimenu(); //查询菜单

void DisplayUimenu(); //显示菜单

void TransferredMethod(); //转存菜单

};

#endif

## 2. 除主函数外的源文件

### 1）手机联系人类的实现 //PhoneContact.cpp

/\*手机联系人类的实现\*/

#include"PhoneContact.h"

#include"Uimenu.h"

PhoneContact::PhoneContact(string rname , string rtel\_num ) {//有参构造函数

name = rname;

tel\_num = rtel\_num;

}

void PhoneContact::modify() {//修改数据成员的值

int n;

Uimenu obj;

change1:{

obj.ChangePhoneUimenu();

cin >> n;

switch (n)

{

case 1: {

cout << "请重新输入联系人姓名：";

cin >> name;

cout << "请重新输入联系人号码：";

cin >> tel\_num;

break;

}

case 2: {

cout << "请重新输入联系人姓名：";

cin >> name;

break;

}

case 3: {

cout << "请重新输入联系人号码：";

cin >> tel\_num;

break;

}

case 0: {

return;

}

default:{

cout << "选择错误，请重新选择！" << endl;

goto change1;

}

}

}

}

istream &operator >>(istream &input, PhoneContact &obj) {//重载>>运算符

cout << "请输入联系人姓名：";

input >> obj.name;

cout << "请输入联系人号码：";

input >> obj.tel\_num;

return input;

}

ostream &operator <<(ostream &output, PhoneContact &obj) {//重载<<运算符

output << "联系人姓名：" << obj.name << "\t";

output << "联系人号码：" << obj.tel\_num << endl;

return output;

}

string PhoneContact::get\_name() { //获取姓名

return name;

}

string PhoneContact::get\_tel\_num() { //获取号码

return tel\_num;

}

void PhoneContact::set(string rname, string rtel\_num) {//为数据成员赋值

name = rname;

tel\_num = rtel\_num;

}

### 2）手机卡联系人类的实现 //CardContact.cpp

/\*手机卡联系人类的实现\*/

#include"cardcontact.h"

#include"Uimenu.h"

CardContact::CardContact(string rname, string rtel\_num, string rhomeland, string rQQ) :PhoneContact(rname, rtel\_num){//有参构造函数

homeland = rhomeland;

QQ = rQQ;

}

void CardContact::modify() {//修改数据成员的值

int n;

Uimenu obj;

change2: {

obj.ChangeCardUimenu();

cin >> n;

switch (n)

{

case 1: {

cout << "请重新输入联系人姓名：";

cin >> name;

cout << "请重新输入联系人号码：";

cin >> tel\_num;

cout << "请重新输入联系人籍贯：";

cin >> homeland;

cout << "请重新输入联系人QQ：";

cin >> QQ;

break;

}

case 2: {

cout << "请重新输入联系人姓名：";

cin >> name;

break;

}

case 3: {

cout << "请重新输入联系人号码：";

cin >> tel\_num;

break;

}

case 4: {

cout << "请重新输入联系人籍贯：";

cin >> homeland;

break;

}

case 5: {

cout << "请重新输入联系人QQ：";

cin >> QQ;

break;

}

case 0: {

return;

}

default: {

cout << "选择错误，请重新选择！" << endl;

goto change2;

}

}

}

}

istream & operator >> (istream & input, CardContact & obj){//重载>>运算符

cout << "请输入联系人姓名：";

input >> obj.name;

cout << "请输入联系人号码：";

input >> obj.tel\_num;

cout << "请输入籍贯：";

input >> obj.homeland;

cout << "请输入QQ：";

input >> obj.QQ;

return input;

}

ostream & operator << (ostream &output, CardContact &obj){//重载<<运算符

output << "联系人姓名：" << obj.name << "\t";

output << "联系人号码：" << obj.tel\_num << "\t";

output << "联系人籍贯：" << obj.homeland << "\t";

output << "联系人QQ：" << obj.QQ << endl;

return output;

}

string CardContact::get\_homeland() { //获取籍贯

return homeland;

}

string CardContact::get\_QQ() { //获取QQ

return QQ;

}

void CardContact::set(string rname, string rtel\_num, string rhomeland, string rQQ) { //为数据成员赋值

PhoneContact::set(rname,rtel\_num);//调用基类set函数

homeland = rhomeland;

QQ = rQQ;

}

### 3）手机通讯簿类的实现 //PhoneMaillist.cpp

/\*手机通讯簿类的实现\*/

#include"phonemaillist.h"

#include"uimenu.h"

#include<fstream>

PhoneMaillist::PhoneMaillist() { //构造函数，读取文本文件中的数据,并根据文件内容创建联系人对象数组

ifstream fin("手机.txt");//打开文件

/\*吸收标题行\*/

char str[100];

fin.getline(str, 100);

fin.getline(str, 100);

fin.getline(str, 100);

fin.getline(str, 100);

phone\_contact\_num = -1;

string rname, rtel\_num;

while (fin >> rname >> rtel\_num )//读取文件中的联系人姓名与号码，且跳过姓名为“新联系人”的联系人

{

PhoneContact temp; //建立手机联系人临时对象

temp.set(rname, rtel\_num); //存储联系人姓名和号码

phone\_contact.push\_back(temp); //压入vector容器

phone\_contact\_num++; //联系人数量+1

}

fin.close();//关闭文件

}

PhoneMaillist::~PhoneMaillist() { //析构函数，将对象数组中的内容写入到文本文件中

ofstream fout("手机.txt");//打开文件

/\*写入文件标题\*/

fout << " ----------------------------------------------- " << endl;

fout << "| 手机联系人 |" << endl;

fout << " ----------------------------------------------- " << endl;

fout << " 姓名\t\t\t号码\t\t\t" << endl;

for (int i = phone\_contact\_num; i >= 0; i--)

{

fout.setf(ios::left); //设置左对齐

fout.width(24); //设置域宽，与上述三个/t对应

fout << phone\_contact[i].get\_name(); //写入姓名

fout.width(24); //设置域宽，与上述三个/t对应

fout << phone\_contact[i].get\_tel\_num() << endl; //写入号码

}

phone\_contact\_num = -1; //待存储联系人数量清空

phone\_contact.clear(); //vector容器清空

fout.close(); //关闭文件

}

void PhoneMaillist::add() { //新建联系人

if (phone\_contact\_num>1000) { //当达到容量上限时不再添加

cout << "手机通讯录容量已满，无法新建联系人！" << endl;

cout << "请尝试在手机卡通讯录新建联系人" << endl;

return;

}

PhoneContact temp; //建立临时对象

cin >> temp; //使用重载 >> 输入数据

phone\_contact.push\_back(temp); //压入vector容器

phone\_contact\_num++; //手机联系人个数+1

cout << "新建联系人成功！" << endl;

}

void PhoneMaillist::Delete(string rtel\_num) { //删除联系人

if (phone\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true) {

bool flag = false;

for (int i = 0; i <= phone\_contact\_num; i++) { //枚举所有联系人

if (phone\_contact[i].get\_tel\_num() == rtel\_num) { //如果号码相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << phone\_contact[i];

cout << "是否删除该联系人？" << endl;

cout << "1.确定删除" << endl;

cout << "2.取消删除" << endl;

int t;

cin >> t;

if (t == 1)

{

phone\_contact.erase(phone\_contact.begin() + i);

phone\_contact\_num--;

cout << "删除成功！" << endl;

}

else //如果取消删除，继续向下枚举

continue;

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入号码？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入号码 :";

cin >> rtel\_num;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，删除完毕！" << endl;

return;

}

}

}

void PhoneMaillist::change(string rtel\_num) { //修改联系人信息

if (phone\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true)

{

bool flag = false;

for (int i = 0; i <= phone\_contact\_num; i++) { //枚举所有联系人

if (phone\_contact[i].get\_tel\_num() == rtel\_num) { //如果号码相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << phone\_contact[i];

cout << "是否修改该联系人？" << endl;

cout << "1.确定修改" << endl;

cout << "2.取消修改" << endl;

int t;

cin >> t;

if (t == 1) {

phone\_contact[i].modify();

cout << "修改成功！" << endl;

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t == 1)

continue;

else

break;

}

else { //如果取消修改，继续向下枚举

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t == 1)

continue;

else

break;

}

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入号码？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入号码 :";

cin >> rtel\_num;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，修改完毕！" << endl;

return;

}

}

}

void PhoneMaillist::inquiry(string rname) { //查询联系人信息

if (phone\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true)

{

bool flag = false;

for (int i = 0; i <= phone\_contact\_num; i++) { //枚举所有联系人

if (phone\_contact[i].get\_name() == rname) { //如果姓名相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << phone\_contact[i];

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t==1)

continue;

else

break;

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入姓名？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入姓名 :";

cin >> rname;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，查询完毕！" << endl;

return;

}

}

}

void PhoneMaillist::display() { //显示所有联系人信息

if (phone\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

cout << "手机通讯录中共有" << phone\_contact\_num + 1 << "个联系人." << endl;

cout << "信息如下：" << endl;

vector <PhoneContact>::iterator temp; //vector迭代器

for (temp = phone\_contact.begin(); temp != phone\_contact.end(); temp++) //显示vector容器内的所有信息

cout << (\*temp);

}

int PhoneMaillist::get\_phone\_contact\_num() { //获取手机通讯录联系人数量

return phone\_contact\_num;

}

void PhoneMaillist::copy(string rname, string rtel\_num) { //将姓名、号码压入动态数组

for (int i = 0; i <= phone\_contact\_num; i++) { //枚举所有联系人

if (phone\_contact[i].get\_name() == rname && phone\_contact[i].get\_tel\_num() == rtel\_num) { //避免存在相同的联系人

cout << "姓名为：" << rname << "　电话为：" << rtel\_num << " 的联系人已存在!" << endl;

return;

}

}

PhoneContact temp; //建立手机联系人临时对象

temp.set(rname, rtel\_num); //将联系人信息赋给临时对象

phone\_contact.push\_back(temp); //压入动态数组

phone\_contact\_num++; //手机联系人数量+1

cout << "已存入姓名为：" << rname << "　电话为：" << rtel\_num << " 的联系人!" << endl;

}

string PhoneMaillist::get\_name(int i) { //获取动态数组第i个联系人的姓名

return phone\_contact[i].get\_name();

}

string PhoneMaillist::get\_tel\_num(int i) { //获取动态数组第i个联系人的号码

return phone\_contact[i].get\_tel\_num();

}

void PhoneMaillist::clear() { //清空手机通讯录

phone\_contact.clear();

phone\_contact\_num = -1;

}

### 4）手机卡通讯簿类的实现 //CardMaillist.cpp

/\*手机卡通讯簿类的实现\*/

#include"cardmaillist.h"

#include"uimenu.h"

#include<fstream>

CardMaillist::CardMaillist() { //构造函数，读取文本文件中的数据,并根据文件内容创建联系人对象数组

ifstream fin("手机卡.txt");//打开文件

/\*吸收标题行\*/

char str[100];

fin.getline(str, 100);

fin.getline(str, 100);

fin.getline(str, 100);

fin.getline(str, 100);

card\_contact\_num = -1;

string rname, rtel\_num,rhomeland,rQQ;

while (fin >> rname >> rtel\_num >> rhomeland >> rQQ)//读取文件中的联系人姓名、号码、籍贯与QQ

{

CardContact temp; //建立手机卡联系人临时对象

temp.set(rname, rtel\_num , rhomeland , rQQ); //存储联系人姓名和号码

card\_contact.push\_back(temp); //压入vector容器

card\_contact\_num++; //联系人数量+1

}

fin.close();//关闭文件

}

CardMaillist::~CardMaillist() { //析构函数，将对象数组中的内容写入到文本文件中

ofstream fout("手机卡.txt");//打开文件

/\*写入文件标题\*/

fout << " ----------------------------------------------------------------------- " << endl;

fout << "| 手机卡联系人 |" << endl;

fout << " ----------------------------------------------------------------------- " << endl;

fout << " 姓名\t\t号码\t\t籍贯\t\tQQ\t\t" << endl;

for (int i = card\_contact\_num; i >= 0; i--)

{

fout.setf(ios::left); //设置左对齐

fout.width(16); //设置域宽，与上述2个/t对应

fout << card\_contact[i].get\_name(); //写入姓名

fout.width(16); //设置域宽，与上述2个/t对应

fout << card\_contact[i].get\_tel\_num(); //写入号码

fout.width(16); //设置域宽，与上述2个/t对应

fout << card\_contact[i].get\_homeland(); //写入籍贯

fout.width(16); //设置域宽，与上述2个/t对应

fout << card\_contact[i].get\_QQ() << endl; //写入QQ

}

card\_contact\_num = -1; //待存储联系人数量清空

card\_contact.clear(); //vector容器清空

fout.close(); //关闭文件

}

void CardMaillist::add() { //新建联系人

if (card\_contact\_num>1000) { //当达到容量上限时不再添加

cout << "手机通讯录容量已满，无法新建联系人！" << endl;

cout << "请尝试在手机卡通讯录新建联系人" << endl;

return;

}

CardContact temp; //建立临时对象

cin >> temp; //使用重载 >> 输入数据

card\_contact.push\_back(temp); //压入vector容器

card\_contact\_num++; //手机联系人个数+1

cout << "新建联系人成功！" << endl;

}

void CardMaillist::Delete(string rtel\_num) { //删除联系人

if (card\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true) {

bool flag = false;

for (int i = 0; i <= card\_contact\_num; i++) { //枚举所有联系人

if (card\_contact[i].get\_tel\_num() == rtel\_num) { //如果号码相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << card\_contact[i];

cout << "是否删除该联系人？" << endl;

cout << "1.确定删除" << endl;

cout << "2.取消删除" << endl;

int t;

cin >> t;

if (t == 1)

{

card\_contact.erase(card\_contact.begin() + i);

card\_contact\_num--;

cout << "删除成功！" << endl;

}

else //如果取消删除，继续向下枚举

continue;

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入号码？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入号码 :";

cin >> rtel\_num;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，删除完毕！" << endl;

return;

}

}

}

void CardMaillist::change(string rtel\_num) { //修改联系人信息

if (card\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true)

{

bool flag = false;

for (int i = 0; i <= card\_contact\_num; i++) { //枚举所有联系人

if (card\_contact[i].get\_tel\_num() == rtel\_num) { //如果号码相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << card\_contact[i];

cout << "是否修改该联系人？" << endl;

cout << "1.确定修改" << endl;

cout << "2.取消修改" << endl;

int t;

cin >> t;

if (t == 1) {

card\_contact[i].modify();

cout << "修改成功！" << endl;

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t == 1)

continue;

else

break;

}

else { //如果取消修改，继续向下枚举

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t == 1)

continue;

else

break;

}

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入号码？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入号码 :";

cin >> rtel\_num;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，修改完毕！" << endl;

return;

}

}

}

void CardMaillist::inquiry(string rname) { //查询联系人信息

if (card\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

while (true)

{

bool flag = false;

for (int i = 0; i <= card\_contact\_num; i++) { //枚举所有联系人

if (card\_contact[i].get\_name() == rname) { //如果姓名相同

flag = true;

cout << "该联系人信息为：" << endl;

cout << card\_contact[i];

cout << "是否寻找相同姓名联系人？" << endl;

cout << "1.确定寻找" << endl;

cout << "2.取消寻找" << endl;

int t;

cin >> t;

if (t == 1)

continue;

else

break;

}

}

if (flag == false) { //如果未找到相应联系人

cout << "在手机通讯录未找到相应联系人!" << endl;

cout << "是否重新输入姓名？" << endl;

cout << "1.重新输入" << endl;

cout << "2.不重新输入" << endl;

int t;

cin >> t;

if (t == 1) {

cout << "请重新输入姓名 :";

cin >> rname;

continue;

}

else {

cout << "输入取消" << endl;

return;

}

}

else { //如果找到相应联系人

cout << "搜索结束，查询完毕！" << endl;

return;

}

}

}

void CardMaillist::display() { //显示所有联系人信息

if (card\_contact\_num<0) { //手机通讯录没有联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

cout << "手机通讯录中共有" << card\_contact\_num + 1 << "个联系人." << endl;

cout << "信息如下：" << endl;

vector <CardContact>::iterator temp; //vector迭代器

for (temp = card\_contact.begin(); temp != card\_contact.end(); temp++) //显示vector容器内的所有信息

cout << (\*temp);

}

int CardMaillist::get\_card\_contact\_num() { //获取手机卡通讯录联系人数量

return card\_contact\_num;

}

void CardMaillist::copy(string rname, string rtel\_num) { //将姓名、号码压入动态数组

for (int i = 0; i <= card\_contact\_num; i++) { //枚举所有联系人

if (card\_contact[i].get\_name() == rname && card\_contact[i].get\_tel\_num() == rtel\_num) { //避免存在相同的联系人

cout << "姓名为：" << rname << "　电话为：" << rtel\_num << " 的联系人已存在!" << endl;

return;

}

}

CardContact temp; //建立手机卡联系人临时对象

temp.set(rname, rtel\_num, "无地址", "无QQ"); //将联系人信息赋给临时对象

card\_contact.push\_back(temp); //压入动态数组

card\_contact\_num++; //手机卡联系人数量+1

cout << "已存入姓名为：" << rname << "　电话为：" << rtel\_num << " 的联系人!" << endl;

}

string CardMaillist::get\_name(int i) { //获取动态数组第i个联系人的姓名

return card\_contact[i].get\_name();

}

string CardMaillist::get\_tel\_num(int i) { //获取动态数组第i个联系人的号码

return card\_contact[i].get\_tel\_num();

}

void CardMaillist::clear() { //清空手机卡通讯录

card\_contact.clear();

card\_contact\_num = -1;

}

### 5）用户类的实现 //User.cpp

/\*用户类的实现\*/

#include"user.h"

#include<iostream>

using namespace std;

User::User() { //构造函数

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

cout << "\* 欢迎使用通讯录管理系统 \*" << endl;

cout << "\* 手机通讯录目前有" << phonemaillist.get\_phone\_contact\_num() + 1 << "个联系人 \*" << endl;

cout << "\* 手机卡通讯录目前有" << cardmaillist.get\_card\_contact\_num() + 1 << "个联系人 \*" << endl;

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

}

User::~User() { //析构函数

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

cout << "\* 感谢您使用本通讯录管理系统 \*" << endl;

cout << "\* 手机通讯录目前有" << phonemaillist.get\_phone\_contact\_num() + 1 << "个联系人 \*" << endl;

cout << "\* 手机卡通讯录目前有" << cardmaillist.get\_card\_contact\_num() + 1 << "个联系人 \*" << endl;

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

/\*显示控制，使界面更友好\*/

system("pause");

system("cls");

}

void User::add() { //新建联系人

change3: {

uimenu.AddUimenu(); //新建菜单

Maillist \*temp; //通讯簿基类指针

int t;

cin >> t;

switch (t) {

case 1: { //新建联系人到手机

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> add(); //调用手机通讯簿对象函数

break;

}

case 2: { //新建联系人到手机卡

temp = &cardmaillist; //指针指向手机卡通讯簿对象

temp -> add(); //调用手机卡通讯簿对象函数

break;

}

case 0: { //返回上一级

return;

}

default: { //选择错误

cout << "选择错误，请重新选择！" << endl;

goto change3;

}

}

}

}

void User::Delete() { //删除联系人

change4: {

uimenu.DeleteUimenu(); //删除菜单

Maillist \*temp; //通讯簿基类指针

string rtel\_num;

int t;

cin >> t;

switch (t)

{

case 1: { //在手机通讯录删除联系人

cout << "请输入要删除的联系人的号码：";

cin >> rtel\_num;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> Delete(rtel\_num); //调用手机通讯簿对象函数

break;

}

case 2: { //在手机卡通讯录删除联系人

cout << "请输入要删除的联系人的号码：";

cin >> rtel\_num;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> Delete(rtel\_num); //调用手机卡通讯簿对象函数

break;

}

case 3: { //在两个通讯录都进行删除

cout << "请输入要删除的联系人的号码：";

cin >> rtel\_num;

cout << "--------------手机通讯录----------------" << endl;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> Delete(rtel\_num); //调用手机通讯簿对象函数

cout << "--------------手机卡通讯录----------------" << endl;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> Delete(rtel\_num); //调用手机卡通讯簿对象函数

break;

}

case 0: { //返回上一级

return;

}

default: { //选择错误

cout << "选择错误，请重新选择！" << endl;

goto change4;

}

}

}

}

void User::change() { //修改联系人信息

change5: {

uimenu.ChangeUimenu(); //修改菜单

Maillist \*temp; //通讯簿基类指针

string rtel\_num;

int t;

cin >> t;

switch (t)

{

case 1: { //在手机通讯录修改联系人

cout << "请输入要修改的联系人的号码：";

cin >> rtel\_num;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> change(rtel\_num); //调用手机通讯簿对象函数

break;

}

case 2: { //在手机卡通讯录修改联系人

cout << "请输入要修改的联系人的号码：";

cin >> rtel\_num;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> change(rtel\_num); //调用手机卡通讯簿对象函数

break;

}

case 3: { //在两个通讯录都进行修改

cout << "请输入要修改的联系人的号码：";

cin >> rtel\_num;

cout << "--------------手机通讯录----------------" << endl;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> change(rtel\_num); //调用手机通讯簿对象函数

cout << "--------------手机卡通讯录----------------" << endl;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> change(rtel\_num); //调用手机卡通讯簿对象函数

break;

}

case 0: { //返回上一级

return;

}

default: { //选择错误

cout << "选择错误，请重新选择！" << endl;

goto change5;

}

}

}

}

void User::inquiry() { //查询联系人信息

change6: {

uimenu.InquiryUimenu(); //查询菜单

Maillist \*temp; //通讯簿基类指针

string rname;

int t;

cin >> t;

switch (t)

{

case 1: { //在手机通讯录查询联系人

cout << "请输入要查询的联系人的姓名：";

cin >> rname;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> inquiry(rname); //调用手机通讯簿对象函数

break;

}

case 2: { //在手机卡通讯录查询联系人

cout << "请输入要查询的联系人的姓名：";

cin >> rname;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp ->inquiry(rname); //调用手机卡通讯簿对象函数

break;

}

case 3: { //在两个通讯录都进行修改

cout << "请输入要查询的联系人的姓名：";

cin >> rname;

cout << "--------------手机通讯录----------------" << endl;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp ->inquiry(rname); //调用手机通讯簿对象函数

cout << "--------------手机卡通讯录----------------" << endl;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp ->inquiry(rname); //调用手机卡通讯簿对象函数

break;

}

case 0: { //返回上一级

return;

}

default: { //选择错误

cout << "选择错误，请重新选择！" << endl;

goto change6;

}

}

}

}

void User::display() { //显示所有联系人信息

change6: {

uimenu.DisplayUimenu(); //显示菜单

Maillist \*temp; //通讯簿基类指针

int t;

cin >> t;

switch (t)

{

case 1: { //显示手机通讯录所有联系人

cout << "--------------手机通讯录----------------" << endl;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> display(); //调用手机通讯簿对象函数

break;

}

case 2: { //显示手机卡通讯录所有联系人

cout << "--------------手机卡通讯录----------------" << endl;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> display(); //调用手机卡通讯簿对象函数

break;

}

case 3: { //显示两个通讯录的所有联系人

cout << "--------------手机通讯录----------------" << endl;

temp = &phonemaillist; //指针指向手机通讯簿对象

temp -> display(); //调用手机通讯簿对象函数

cout << "--------------手机卡通讯录----------------" << endl;

temp = &cardmaillist; //指针指向手机通讯簿对象

temp -> display(); //调用手机卡通讯簿对象函数

break;

}

case 0: { //返回上一级

return;

}

default: { //选择错误

cout << "选择错误，请重新选择！" << endl;

goto change6;

}

}

}

}

void User::move\_card\_to\_phone() { //移动手机卡通讯录到手机通讯录

if (cardmaillist.get\_card\_contact\_num()<0) { //手机卡通讯录无联系人时

cout << "手机卡通讯录无联系人！" << endl;

return;

}

for (int i = 0; i <= cardmaillist.get\_card\_contact\_num(); i++) { //枚举手机卡通讯录所有联系人

if (phonemaillist.get\_phone\_contact\_num() >= 1000) { //当手机通讯录容量满时

cout << "手机通讯录已满！无法增加新联系人！" << endl;

return;

}

phonemaillist.copy(cardmaillist.get\_name(i), cardmaillist.get\_tel\_num(i)); //将手机卡通讯录的第i个用户存入手机通讯录

}

cardmaillist.clear(); //清空手机卡通讯录

cout << "移动成功！" << endl;

}

void User::move\_phone\_to\_card() { //移动手机通讯录到手机卡通讯录

if (phonemaillist.get\_phone\_contact\_num()<0) { //手机通讯录无联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

for (int i = 0; i <= phonemaillist.get\_phone\_contact\_num(); i++) { //枚举手机通讯录所有联系人

if (cardmaillist.get\_card\_contact\_num() >= 1000) { //当手机卡通讯录容量满时

cout << "手机卡通讯录已满！无法增加新联系人！" << endl;

return;

}

cardmaillist.copy(phonemaillist.get\_name(i), phonemaillist.get\_tel\_num(i)); //将手机通讯录的第i个用户存入手机卡通讯录

}

phonemaillist.clear(); //清空手机通讯录

cout << "移动成功！" << endl;

}

void User::copy\_card\_to\_phone() { //复制手机卡通讯录到手机通讯录

if (cardmaillist.get\_card\_contact\_num()<0) { //手机卡通讯录无联系人时

cout << "手机卡通讯录无联系人！" << endl;

return;

}

for (int i = 0; i <= cardmaillist.get\_card\_contact\_num(); i++) { //枚举手机卡通讯录所有联系人

if (phonemaillist.get\_phone\_contact\_num() >= 1000) { //当手机通讯录容量满时

cout << "手机通讯录已满！无法增加新联系人！" << endl;

return;

}

phonemaillist.copy(cardmaillist.get\_name(i), cardmaillist.get\_tel\_num(i)); //将手机卡通讯录的第i个用户存入手机通讯录

}

cout << "复制成功！" << endl;

}

void User::copy\_phone\_to\_card() { //复制手机通讯录到手机卡通讯录

if (phonemaillist.get\_phone\_contact\_num()<0) { //手机通讯录无联系人时

cout << "手机通讯录无联系人！" << endl;

return;

}

for (int i = 0; i <= phonemaillist.get\_phone\_contact\_num(); i++) { //枚举手机通讯录所有联系人

if (cardmaillist.get\_card\_contact\_num() >= 1000) { //当手机卡通讯录容量满时

cout << "手机卡通讯录已满！无法增加新联系人！" << endl;

return;

}

cardmaillist.copy(phonemaillist.get\_name(i), phonemaillist.get\_tel\_num(i)); //将手机通讯录的第i个用户存入手机卡通讯录

}

cout << "复制成功！" << endl;

}

### 6）菜单类的实现 //Uimenu.cpp

/\*菜单类的实现\*/

#include"uimenu.h"

void Uimenu::MainUimenu() { //主菜单

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

cout << "\* 1.新建联系人 \*" << endl;

cout << "\* 2.删除联系人 \*" << endl;

cout << "\* 3.查询联系人 \*" << endl;

cout << "\* 4.修改联系人信息 \*" << endl;

cout << "\* 5.浏览所有联系人信息 \*" << endl;

cout << "\* 6.将联系人的信息从手机转存到手机卡上 \*" << endl;

cout << "\* 7.将联系人的信息从手机卡转存到手机上 \*" << endl;

cout << "\* 0.退出 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::AddUimenu() { //添加联系人菜单

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

cout << "\* 1.新建联系人到手机 \*" << endl;

cout << "\* 2.新建联系人到手机卡 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::DeleteUimenu() { //删除菜单

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

cout << "\* 1.在手机通讯录删除联系人 \*" << endl;

cout << "\* 2.在手机卡通讯录删除联系人 \*" << endl;

cout << "\* 3.在两个通讯录都进行删除 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::ChangeUimenu() { //修改菜单

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

cout << "\* 1.在手机通讯录修改联系人 \*" << endl;

cout << "\* 2.在手机卡通讯录修改联系人 \*" << endl;

cout << "\* 3.在两个通讯录都进行修改 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::ChangePhoneUimenu() { //修改手机联系人菜单

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

cout << "\* 1.修改联系人姓名和号码 \*" << endl;

cout << "\* 2.修改联系人姓名 \*" << endl;

cout << "\* 3.修改联系人号码 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::ChangeCardUimenu() { //修改手机卡联系人菜单

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

cout << "\* 1.修改联系人全部信息 \*" << endl;

cout << "\* 2.修改联系人姓名 \*" << endl;

cout << "\* 3.修改联系人号码 \*" << endl;

cout << "\* 4.修改联系人籍贯 \*" << endl;

cout << "\* 5.修改联系人QQ \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::InquiryUimenu() { //查询菜单

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

cout << "\* 1.在手机通讯录查询联系人 \*" << endl;

cout << "\* 2.在手机卡通讯录查询联系人 \*" << endl;

cout << "\* 3.在两个通讯录都进行查询 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::DisplayUimenu() { //显示菜单

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

cout << "\* 1.显示手机通讯录所有联系人 \*" << endl;

cout << "\* 2.显示手机卡通讯录所有联系人 \*" << endl;

cout << "\* 3.显示两个通讯录的所有联系人 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

void Uimenu::TransferredMethod() { //转存菜单

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

cout << "\* 1.复制 \*" << endl;

cout << "\* 2.移动 \*" << endl;

cout << "\* 0.返回上一级 \*" << endl;

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

cout << "请选择要进行的操作：";

}

## 3. 主函数源文件//Main.cpp

/\*主函数\*/

#include"uimenu.h"

#include"phonecontact.h"

#include"cardcontact.h"

#include"maillist.h"

#include"phonemaillist.h"

#include"cardmaillist.h"

#include"user.h"

int main()

{

User user; //创建用户对象

Uimenu uimenu; //创建菜单对象

int m=1;

while (m != 0) {

change\_begin: {

/\*显示控制，使界面更友好\*/

system("pause");

system("cls");

uimenu.MainUimenu(); //显示主菜单

cin >> m;

switch (m) { //对主菜单操作选择

case 1: { //新建联系人

user.add();

goto change\_begin;

}

case 2: { //删除联系人

user.Delete();

goto change\_begin;

}

case 3: { //查询联系人

user.inquiry();

goto change\_begin;

}

case 4: { //修改联系人信息

user.change();

goto change\_begin;

}

case 5: { //浏览所有联系人信息

user.display();

goto change\_begin;

}

case 6: { //将联系人的信息从手机转存到手机卡上

Transferred1: {

uimenu.TransferredMethod();

int n;

cin >> n;

switch (n) {

case 1: {

user.copy\_phone\_to\_card();

goto change\_begin;

}

case 2: {

user.move\_phone\_to\_card();

goto change\_begin;

}

case 0: {

goto change\_begin;

}

default: {

cout << "选择错误，请重新选择！" << endl;

goto Transferred1;

}

}

}

}

case 7: { //将联系人的信息从手机卡转存到手机上

Transferred2: {

uimenu.TransferredMethod();

int n;

cin >> n;

switch (n) {

case 1: {

user.copy\_card\_to\_phone();

goto change\_begin;

}

case 2: {

user.move\_card\_to\_phone();

goto change\_begin;

}

case 0: {

goto change\_begin;

}

default: {

cout << "选择错误，请重新选择！" << endl;

goto Transferred2;

}

}

}

}

case 0: { //退出

/\*显示控制，使界面更友好\*/

system("pause");

system("cls");

break;

}

default: {

cout << "选择错误，请重新选择！" << endl;

goto change\_begin;

}

}

}

}

getchar();

return 0;

}