数据结构与算法

实验报告（一）

实验题目：C++环境下简单通讯录的实现

姓 名： 魏靖

学 号： 2021213513

日 期： 2022.03.27

自我评分： 【 A 】90

自我评分说明：A+，A，B+，B，B-，C，D，分别对应分数95、90、85、80、75、70、60

诚信声明

本人郑重承诺：本实验程序和实验报告均是本人独立学习和工作所获得的成果。尽我所知，实验报告中除特别标注的地方外，不包含其他同学已经发表或撰写过的成果；实验程序中对代码工作的任何帮助者所作的贡献均做了明确的说明，并表达了谢意。

如有抄袭，本人原因承担因此而造成的任何后果。

特此声明。

签名：魏靖 日期：2022.03.27

程序引用说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 引用项 | 来源 | 相同代码行数 |
| 1 | 查找函数 | 《书名》 |  |
| 2 | 构造有序链表函数 | 互联网网址 |  |
| 3 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 小计 | | |  |

总代码行数\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; 引用占比\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1、实验简介

建立通讯录存储联系人信息，实现通过姓名查找进行删除修改等操作，实现通讯录一览，然后可以展现单个人物的全部信息，包括ID、姓名、性别、电话、地址（街道、城市、国家）信息，考虑过建立分类，但感觉如果采用树型结构会更好，便没有往下做去。

2、程序框架

建立节点

PhoneList类private成员

Public函数：

People\* loadList();//节点链接到链表（头插法）

void createList(People\*);//节点数据录入

int deleteList(string);//删除节点

People\* searchList(string);//查找节点

void printList();//打印联系人列表

void printPeople(People\*);//打印联系人详细信息

void changePeople(People\*);//修改联系人信息

void MENU();

void Listfile();//写入文件（要求先在同目录中存入myphonelist.txt）

void menu();//主菜单

People\* Copy(People\*);//导入数据节点

P.S. include<stdlib.h>主要便于后续使用atoi函数

1. **关键代码实现**

**3.1文件输入输出流**

从实验之初，就决定要想办法使通讯录保存下来，便复习上学期C++文件输入输出流的相关知识，在整理好链表全部功能之后探索输出，最后探索输入（因为最开始考虑添加联系人先设定添加n个，然后循环输入，但在一开始的实操过程中没有实现指针的存储，所以后面换成了一个一个输入，时间复杂度也没有变话，然而在输入文件的时候就不得不再用循环了，再探索清楚后，也明白了之前为什么循环不能成功输入，更深一步理解了链表存储过程）

具体思路：

**3.1.1 输出**

打开文件流

建立指针指向当前front指针所指的节点（数据全部初始化）

指针不断前移

只要该指针的prior不为空

就输出数据

到最后会输出一项多余默认数据

（整体完善：1、输出共几位联系人2、在输入时不输入此多余数据）

**3.1.2输入**

打开文件流

读取文件数据，分别存进节点中

节点与链表产生链接

因为ID默认为0，因而当读取到文件name为“0”时跳出循环

完成链接展现菜单

**3.2双向链表的实现**

参考书上在两节点之间插入节点的方法，采用头插法建立节点间联系，同时借用网上定义pn的做法，防止后续出现未定义情况，之前代码不明所以的错误得到了解决。

**3.3异常处理**

4、不足

（1）要求使用的char结构输入输出修改比较复杂后面就放弃了。

（2）一开始想的实现class把数据保护起来，但在实际操作的时候先是遇到析构函数的处理问题，后面又涉及到与char\*结合的访问权限，想过用友元函数去改，但是思路理下来更复杂了，又放弃了。

（3）对于变量的初始化和始终初始化还是难以区分，需要依靠编译器提示，最好做法应是定义即初始化，用完即delete。

（4）有意避免了使用template，因为调试了很久template会出现问题，后来解决了之后编译器突然又开始报错，就更新了之前的思路，没有再纠结于那上面，导致至今class T那种用法还是只会书上那种，迁移不了，不知道怎么正确迁移到自定义类型上来。

5、心得体会

（1）析构函数不能作为私有成员。

（2）定义class A返回值为T\*的函数，在声明后的定义形式为T\*A::函数名(){}而不是A在前。

（3）异常处理的核心在于保护语句异常条件，且必须有异常捕获处理分支，在多次cin>>，如果我本该输入int却输入了char，程序会忽略中间所有的cin跳到要输入char的地方，把输入的char当做是那时候输入的，但是在后续处理过程中会引发重复异常，导致同一输出语句反复霸屏无休止。

（4）双向链表头插法插入节点时p->next->prior=p放最后，不然指针会出现空的异常状况，导致无法读取，链接失败，通过实操进一步理解了节点之间建立链接的机制，也更好地掌握了利用断点分析程序错误不断修改的方法。

**代码：**

#include<iostream>

#include<string>

#include<fstream>

#include <stdlib.h>

using namespace std;

typedef struct Node

{

int ID=0;

string Name;

char ch='b';

string phone;

string Street;

string City;

string State;

struct Node\* next=NULL;

struct Node\* prior=NULL;

}People;

class PhoneList {

private:

People f= { 0,"0",'b',"0","0","0","0",NULL,NULL}; //头节点初始化

People l = { 0,"0",'b',"0","0","0","0",NULL,NULL };//辅助节点初始化（防止指向空，或未定义）

People\* front = &f; //设置头指针，指向头节点

public:

People\* loadList();//节点链接到链表（头插法）

void createList(People\*);//节点数据录入

int deleteList(string);//删除节点

People\* searchList(string);//查找定位到节点

void printList();//打印联系人列表

void printPeople(People\*);//打印联系人详细信息

void changePeople(People\*);//改变联系人信息

void MENU();

void Listfile();//写入文件（要求先在同目录中存入myphonelist.txt）

void menu();//主菜单

People\* Copy(People\*);//导入数据节点

};

People\* PhoneList::loadList()

{

People\* p = new People; //给这个新节点分配空间

createList(p);

p->next = front->next; //p的下一个指向头指针指向节点的下一个

front->next = p; //头指针指向节点的下一个指向p

p->prior = front; //p的上一个指向指针指向节点的下一个

p->next->prior = p;

front = p; //头指针指向p

return p;

}

void PhoneList::createList(People\* P)

{

cout << "输入ID" << endl;

string a;

int x = 0;

try {

cin >> a;

for (int i = 0; i < a.length(); i++) {

char s = a[i];

if (!(s >= '0' && s <= '9'))

throw a;

x = atoi(a.c\_str());

}

}

catch (string s) {

cout << "ID中请勿包含字符,已默认ID=0" << endl;

}

P->ID = x; //异常处理，防止输入类型有误

string name, mobile;

cout << "输入姓名：";

cin >> name;

P->Name = name;

cout << "输入电话：";

cin >> mobile;

P->phone = mobile;

cout << "输入性别：‘b’男，‘g’女" << endl;

char sex;

cin >> sex;

while (sex != 'b'&&sex != 'g') {

cout << "输入错误，‘b’男，‘g’女" << endl;

cin >> sex;

}

P->ch = sex;

string street;

string city;

string state;

cout << "街道：";

cin >> street;

P->Street = street;

cout << "城市：";

cin >> city;

P->City = city;

cout << "国家：";

cin >> state;

P->State = state;

}

int PhoneList::deleteList(string key\_name)

{

People\* p;

p = &l;

for (p; p->prior != NULL; p = p->prior)

{

if (p->Name == key\_name)

{

printPeople(p);

cout << "确定？Y/N" << endl;

char judge;

cin >> judge;

if (judge == 'Y') {

front = l.prior;

p->prior->next = p->next; //p的上一个的下一个指向p的下一个

p->next->prior = p->prior; //p的下一个的上一个指向p的上一个

delete p; //释放p

}

else;

return 0; //删除后退出

}

}

cout << "没有此人！！！！" << endl;

return 0;

}

People\* PhoneList::searchList(string key)

{

People\* p;

p = &l;

for (p; p->prior != NULL; p = p->prior)

{

if (p->Name == key)

{

return p; //返回相应节点的指针

}

}

return NULL;

}

void PhoneList::printList()

{

People\* p;

p = front;

int i = 1; //显示序号

while (p->prior )

{

cout << " " << i << ": " << p->Name << endl; //打印p指针对应节点的名字

p = p->prior; //指针指向上一个节点

i++;

}

cout << "‘1’查看联系人，‘0’返回" << endl;

int t;

cin >> t;

while (t) {

cout << "输入联系人名字" << endl;

string key;

cin >> key;

printPeople(searchList(key));

cout << "继续查看？1/0" << endl;

cin >> t;

}

}

void PhoneList::printPeople(People\* P)

{

if (!P)

{

cout << "没有此人！！ " << endl;

}

else {

cout << "ID:" << P->ID << endl;

cout << " 姓名: " << P->Name << endl;

cout << "性别:";

if (P->ch == 'b')cout << "男" << endl;

else cout << "女" << endl;

cout << " 电话: " << P->phone << endl;

cout << " 街道: " << P->Street << endl;

cout << " 城市: " << P->City << endl;

cout << " 国家:" << P->State << endl;

}

}

void PhoneList::changePeople(People\* P) {

printPeople(P);

cout << "你想改什么？‘1’ID‘2’姓名‘3’电话‘4’地址‘5’性别‘0’返回" << endl;

int t, m=0;

cin >> t;

do {

switch (t) {

case 1:{

cout << "输入新ID" << endl;

string a;

int x = 0;

try {

cin >> a;

for (int i = 0; i < a.length(); i++) {

char s = a[i];

if (!(s >= '0' && s <= '9'))

throw a;

x = atoi(a.c\_str());

}

}

catch (string s) {

cout << "ID中请勿包含字符,已默认ID=0" << endl;

}

P->ID = x; //异常处理

}

case 2:{

string name;

cout << "输入姓名：";

cin >> name;

P->Name = name;

}

case 3:{

string mobile;

cout << "输入电话：";

cin >> mobile;

P->phone = mobile;

}

case 4:{

string street;

string city;

string state;

cout << "街道：";

cin >> street;

P->Street = street;

cout << "城市：";

cin >> city;

P->City = city;

cout << "国家：";

cin >> state;

P->State = state;

}

case 5: {

if (P->ch == 'b')P->ch = 'g';

else P->ch = 'b';

}

case 0: {m = 0; break; }

default:break;

}

if (!m)break;

cout << "‘1’继续修改‘0’返回" << endl;

cin >> m;

} while (m);

}

void PhoneList:: MENU()

{

cout << " 1.添加 2.删除 3.查找 4.查看通讯录 5.修改联系人 6.更新txt文档 " << endl;

}

void PhoneList::Listfile()

{

People\* p; //定义结构体类型的指针

p = &l; // 让p 等于当前front指针所指的节点

int i = 0; //记录个数

ofstream fout("myphonelist.txt");

while (p->prior )

{

p = p->prior; //指针指向上一个节点

fout << p->ID << endl;

fout << " " << p->Name << endl;

fout << " " << p->phone << endl;

if(p->ch=='b')fout << " 男" << endl;

else fout << " 女" << endl;

fout << " " << p->Street << endl;

fout << " " << p->City << endl;

fout << " " << p->State << endl;

fout << endl;

i++;

}

fout << "共" << i - 1 << "位联系人" << endl;

fout.close(); //关闭文件流

}

void PhoneList::menu()

{

cout << "welcome to use!是否保留并读取myphonelist数据？Y/N " << endl;

char x='N';

int i = 0;

string key;

f.next = &l; //初始化头尾节点并链接

l.prior = &f;

cin >> x;

if (x == 'Y') {

ifstream f("myphonelist.txt");

while (f) {

People\* p = new People;

int x = 0;

f >> x;

p->ID = x;

string name, mobile;

f >> name;

p->Name = name;

if (p->Name == "0")break;

f >> mobile;

p->phone = mobile;

string sex;

f >> sex;

if (sex == "男")p->ch = 'b';

else p->ch = 'g';

string street;

string city;

string state;

f >> street;

p->Street = street;

f >> city;

p->City = city;

f >> state;

p->State = state;

Copy(p);

}

}

else;

cout << "1.添加 2.删除 3.查找 4.查看通讯录 5.修改联系人 " << endl;

while (1)

{

cin >> i;

switch (i) {

case 1:

loadList(); //新建节点并插入链表

cout << "添加完成！！！" << endl;

MENU();

break;

case 2:

cout << "请输入名字：";

cin >> key;

deleteList(key); //删除节点

MENU();

break;

case 3:

cout << "请输入名字：";

cin >> key;

printPeople(searchList(key));

MENU();

break;

case 4:

printList(); //打印联系人列表

MENU();

break;

case 5:

cout << "请输入名字：";

cin >> key;

changePeople(searchList(key));

MENU();

break;

case 6:

Listfile(); //把所有节点的信息更新到"myphonelist.txt"文件

break;

default:

break;

}

}

}

People\*PhoneList:: Copy(People\*p) {

p->next = front->next; //p的下一个指向头指针指向节点的下一个

front->next = p; //头指针指向节点的下一个指向p

p->prior = front; //p的上一个指向指针指向节点的下一个

p->next->prior = p;

front = p; //头指针指向p

return p;

}

int main() {

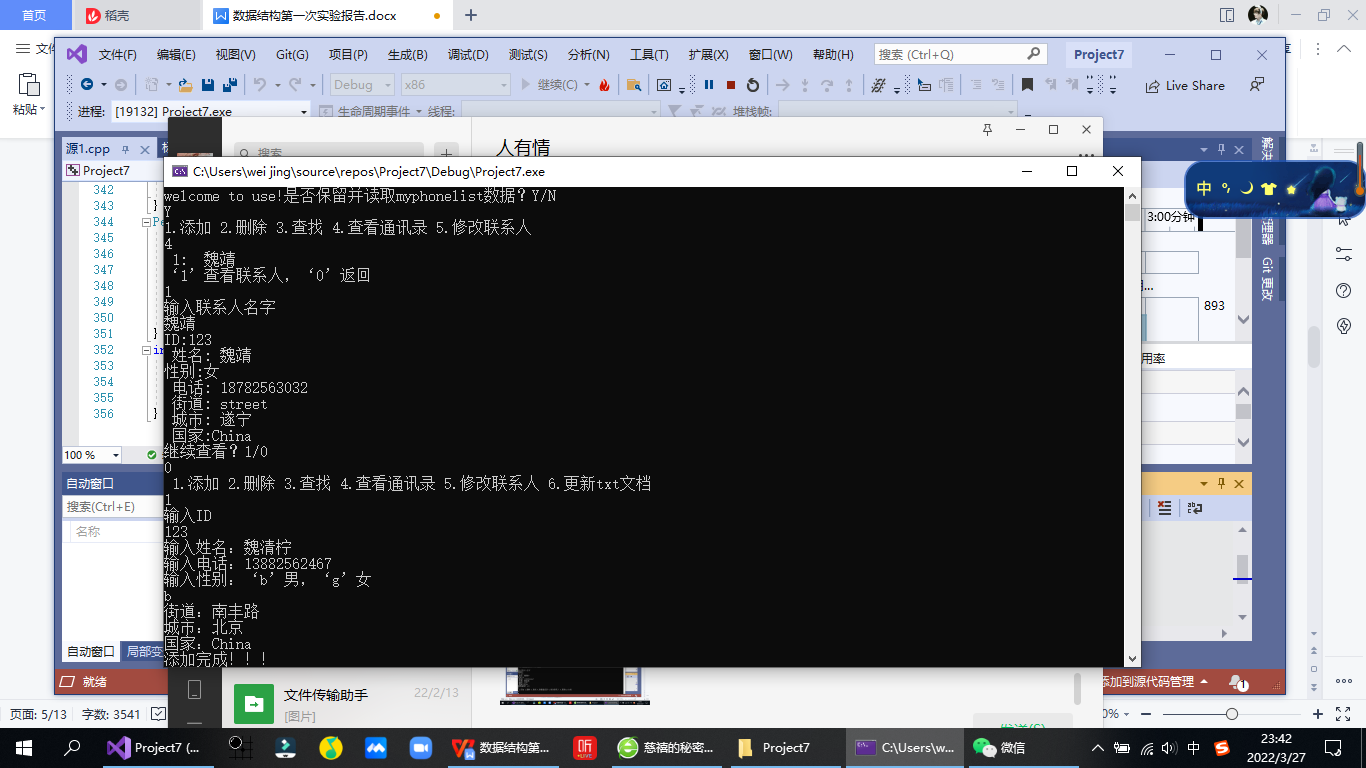
PhoneList MyPhoneList;

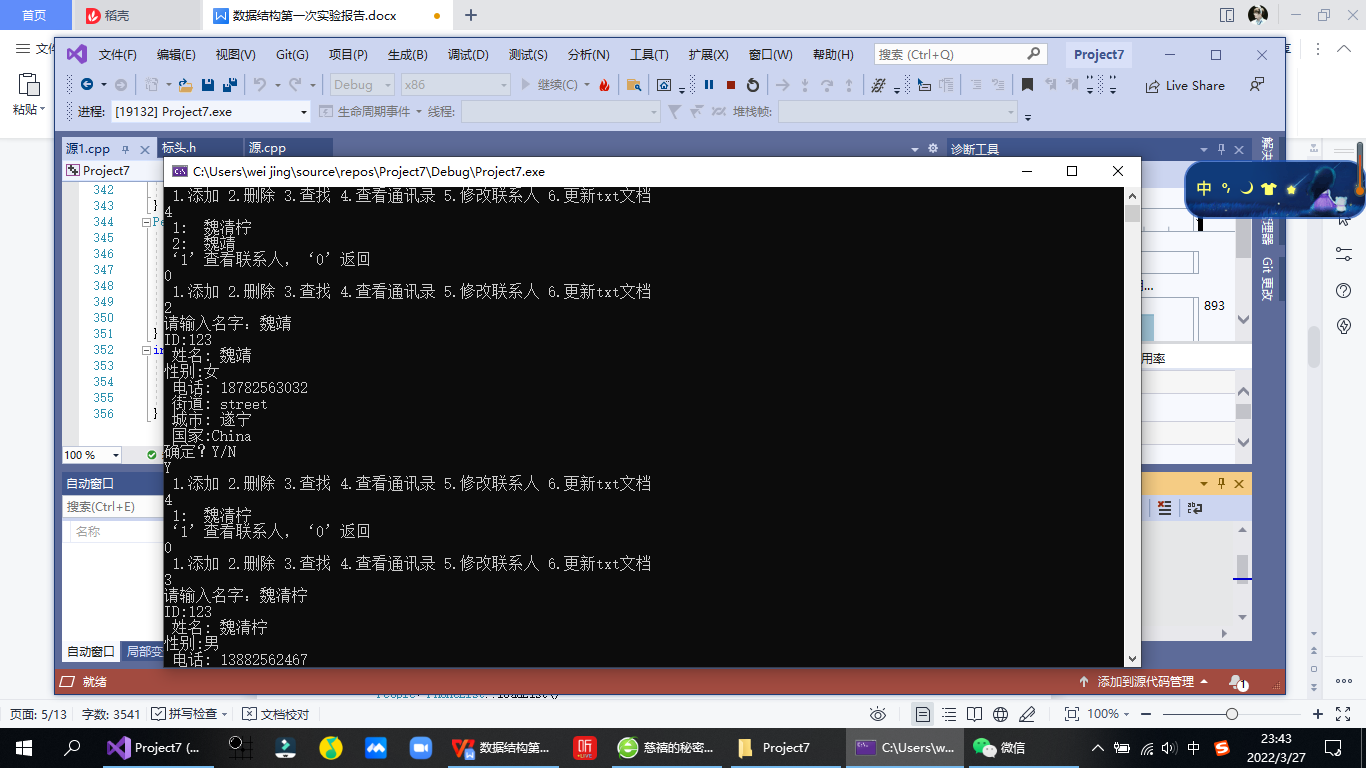
MyPhoneList.menu();

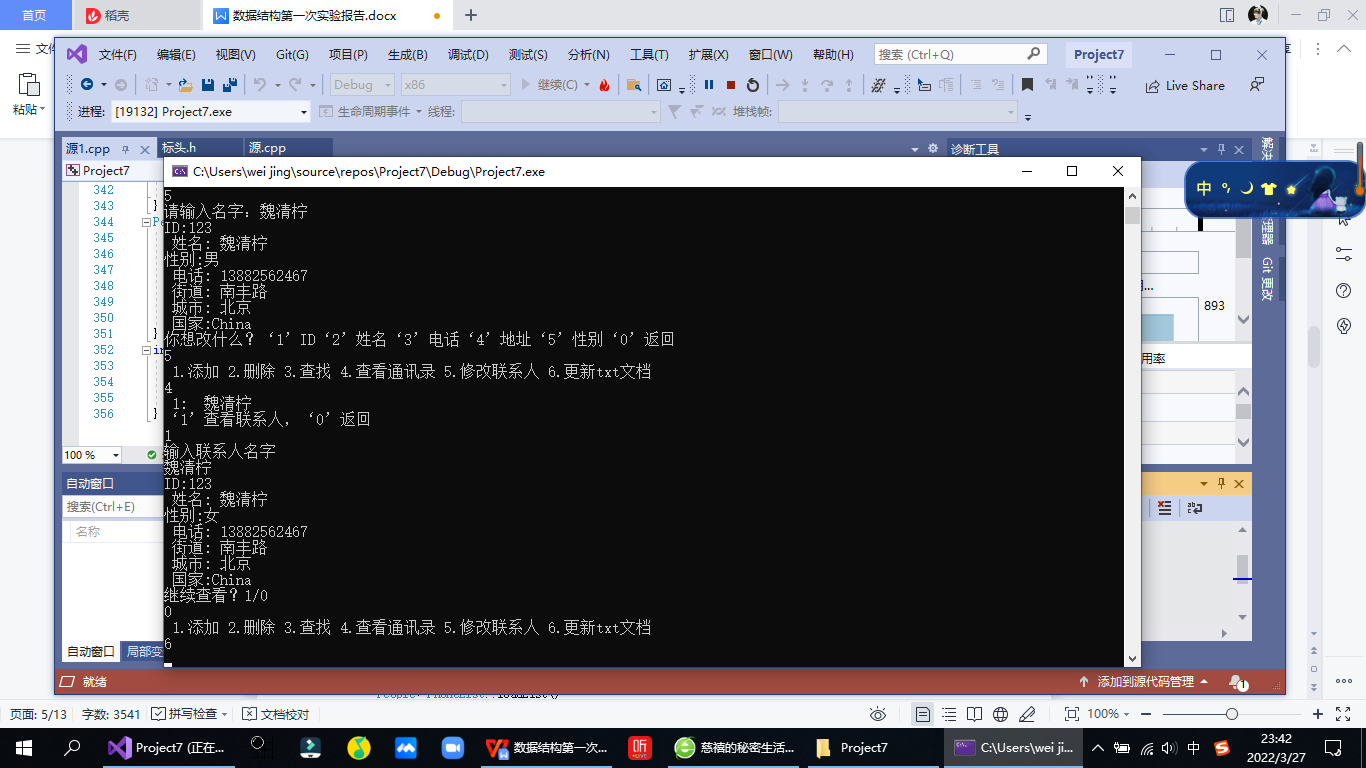
return 0;

}

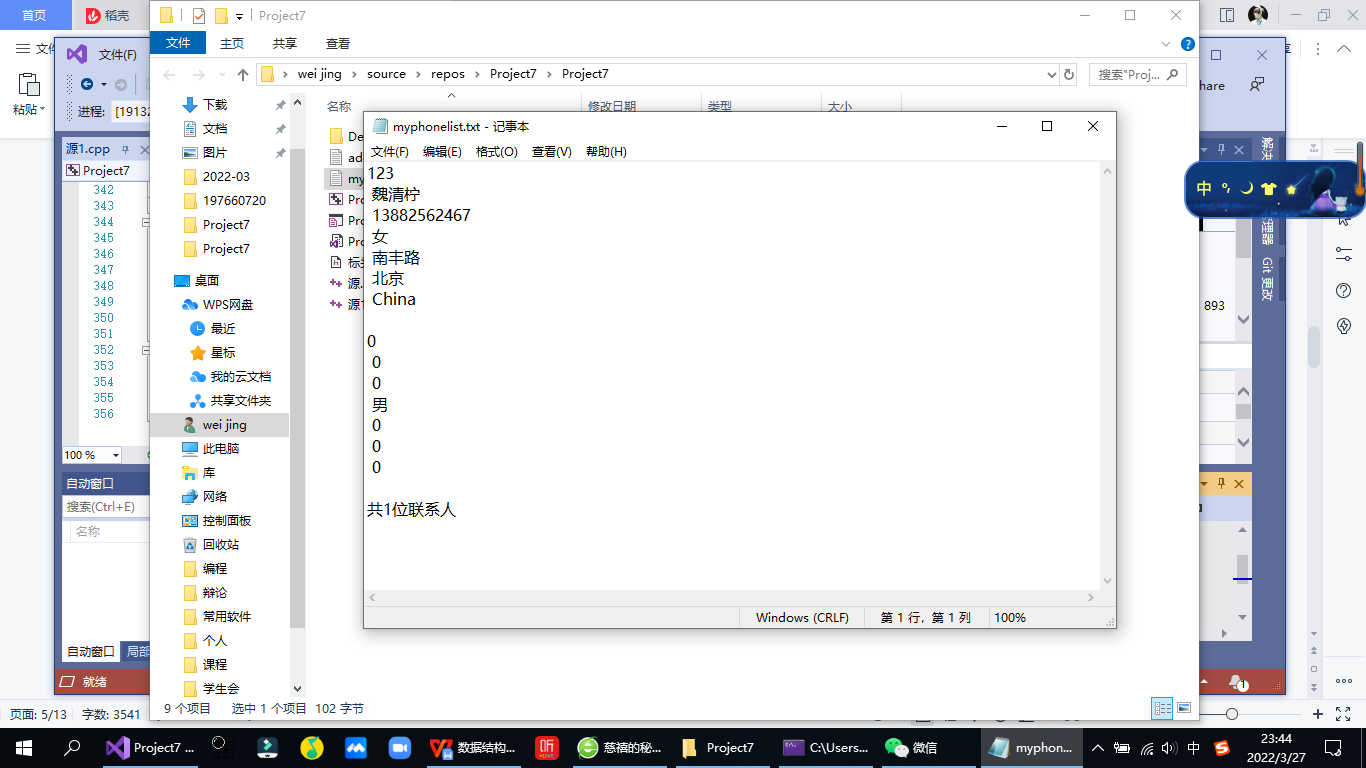
**调试各功能试用结果：**

****

****

****

**运行后的记事本：**

****