东北大学秦皇岛分校

计算机与通信工程学院

School of Computer and Communication Engineering

课程设计报告

通讯录管理系统

学　　院	计算机与通信工程学院
专　　业	电子信息类
班级序号	180409
学　　号	**20188068**
姓　　名	孔天欣
指导教师	程龙
验收日期	2019 年 1 月 11 日

核心知识点清单，由学生确认

数组	链表	指针	文件读写	默认参数	函数模板	多文件	类	派生	虚函数	友元函数	重载	多继承
√		√	√			√	√					

以下为教师评分表，学生不可填写

程序质量 （60%）	课程设计报告 （20%）	答辩效果 （20%）	总分	等级

评分标准与说明：

1、 程序质量（60%）包含程序正确性与所用知识点数量（40%），代码可读性（10%）与界面友好性（10%）。

2、 课程设计报告（20%）要求排版规范，模块设计有文字说明，图、表、代码清单要有序号和名字。

3、 现场答辩（20%）要求根据学生制作的 PPT、讲述清晰、回答问题等情况综合评分。

4、 收齐所有纸质报告的同时，要求学委收集所有学生的代码工程、报告电子版和答辩 PPT 以备存档。

目录

通讯录管理系统

1. 课程设计题目及要求

建立一个通讯录管理系统，能够完成对通讯录的简单管理。用户通过输入序号的方式，来执行相应的功能。

通讯录中需要包含人员的编号、姓名、性别、通讯地址、邮箱地址、电话等信息和对相关信息进行操作的方法。

2 基本功能概述

2.1 基本功能

通讯录管理系统包含通讯录的大多数基本功能，本管理系统能够实现添加功能和删除功能，进行新的通讯录信息的录入和删除；能够通过查询功能，以不同的方式进行指定通讯录人员信息的查询；可以通过显示功能，来获得本通讯录中所有人员的信息；通过编辑功能，对通讯录人员的指定信息进行修改；保存和读取功能能够将通讯录内所有人员信息储存到本地文件中，并可进行读取操作。

2.2 知识点清单

该通讯录管理系统主要应用到以下知识点：

1、运用类有关的知识，建立了通讯录类，并在该类内声明了通讯录人员的数据成员以及各种操作功能的成员函数。

2、运用数组和对象有关的知识，建立了对象数组以储存通讯录人员的信息。

3、部分成员函数对数组下标（book_num）进行了引用，以确定下标的实时变化，以及运用下标实现添加、删除、查询等功能。

4、部分成员函数（如 book_check()）运用了指向对象的指针，用来检验对象信息格式的正确性。

5、部分成员函数（如 book_save()）运用了外部文件的储存和读取相关知识。

6、对于用户输入错误数据类型的问题，采用了清除缓冲区的有关知识。

3 设计思路

通过模块化的编程方式，将所有功能分为不同模块，并通过输入数字序号(1~8)的方式来执行不同模块，从而实现相应的功能。

在本通讯录管理系统中，输入相应的序号可以达成以下功能：

输入 1：添加通讯录信息

输入 2：查询指定通讯录信息

输入 3：查询全部通讯录信息

输入 4：删除指定信息

输入 5：编辑指定信息

输入 6：保存通讯录

输入 7：读取通讯录

输入 8：退出系统

具体流程图和模块结构图以及工程结构组织图如图 3-1、图 3-2、图 3-3 所示：

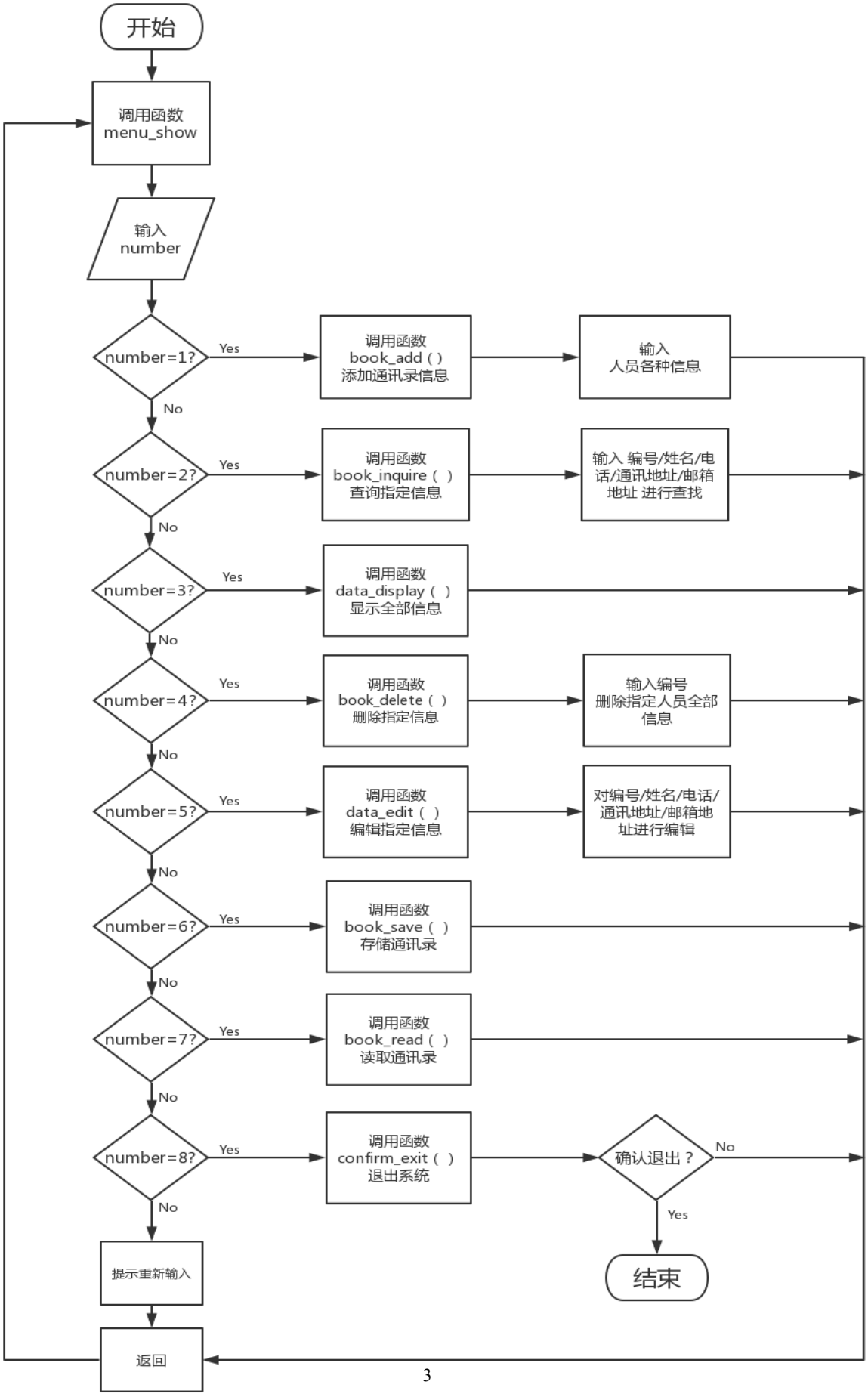

开始
调用函数
menu_show
输入
number
number=1?
Yes
调用函数
book_add ( )
添加通讯录信息
输入
人员各种信息
No
number=2?
Yes
调用函数
book_inquire ( )
查询指定信息
输入 编号/姓名/电话/通讯地址/邮箱地址 进行查找
No
number=3?
Yes
调用函数
data_display ( )
显示全部信息
No
number=4?
Yes
调用函数
book_delete ( )
删除指定信息
输入编号
删除指定人员全部信息
No
number=5?
Yes
调用函数
data_edit ( )
编辑指定信息
对编号/姓名/电话/通讯地址/邮箱地址进行编辑
No
number=6?
Yes
调用函数
book_save ( )
存储通讯录
No
number=7?
Yes
调用函数
book_read ( )
读取通讯录
No
number=8?
Yes
调用函数
confirm_exit ( )
退出系统
确认退出？
No
Yes
结束
No
提示重新输入
返回

图 3-1 流程图

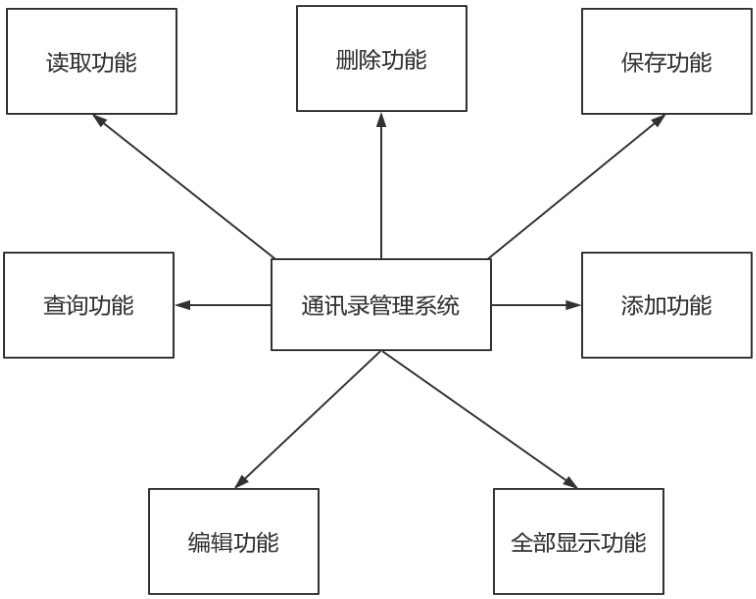


图 3-2 模块结构图

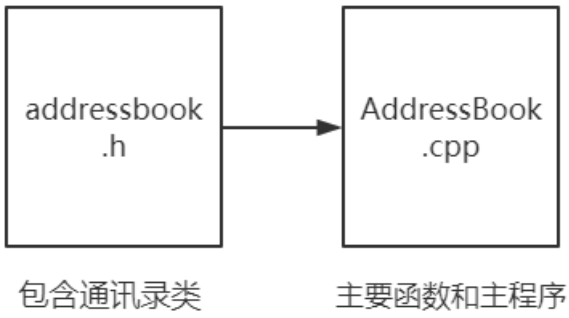


图 3-3 工程结构组织图

4 程序设计

4.1 设计步骤

先建立通讯录类。然后在主函数内建立 switch 分支结构，用此结构来根据输入的不同序号实现相应的功能。之后根据需要一步步在 switch 中扩展分支，在不同分支中添加功能以及相关函数，逐步完成整个程序的编写。

4.2 关键功能

1、采用了类以及对象数组对通讯录人员进行储存的方式。
2、实现了增加、删除、查询、保存、读取等基本功能。

3、增加了对象的各项数据的正确性检查功能。
4、能够修复用户输入错误类型数据导致系统无法正常运行的问题。
5、运用了模块化编程的思想，使代码简洁易懂，层次清晰。
6、图形化界面以及适当的文字引导，尽可能增加对用户的友好性。

4.3 关键功能的实现

1） 定义通讯录类，声明相关数据成员和相关成员函数。

```
class address_book                                    //定义通讯录类
{
private:
    int number;                                       //编号
    char name[20];                                    //姓名
    char sex[20];                                     //性别
    char mailing_address[50];                         //通讯地址
    char email_address[50];                           //邮箱地址
    char phone_number[50];                            //电话号码
public:
    void book_add(int &book_num);                     //添加功能
    static void data_display(int book_num);           //全部展示功能
    static void book_inquire(int book_num);           //查询功能
    static void book_delete(int &book_num);           //删除功能
    static void data_edit(int &book_num);             //编辑功能
    static void book_save(int book_num);              //保存功能
    static void book_read(int &book_num);             //读取功能
    bool book_check(address_book *book);              //格式检查
}book[100];                                           //通讯录数组
```

私有数据成员中包含了通讯录成员的各项信息，公用成员函数中则是各项功能的声明。使用静态类成员函数是为了能在主函数中直接调用函数，不需要通过对象的方式，因为调用过程本身不需要经过对象，也不需要对对象进行相关操作，如果通过对象调用可能会造成对代码的误解，通过设置静态成员函数的方式，使主函数中通过类名调用成员函数，从而使代码可读性增强。

2）模块化编程

通过采用 switch 结构的方式，在输入不同数字后即可调用不同函数，从而实现对应的有关功能，也可以使主函数简洁明了，直观易懂。

```
    while (1)                                          //反复循环，来重复输入。
    {
        menu_show();
        while (!(cin >> number))                        //输入错误号码
        {
            cout << "请输入 1 到 8 内的数字" << endl;
            cin.clear();
            cin.sync();
            while (cin.get() != '\n')
            {
                continue;
            }
        }                                              //已经输入正确号码
        switch  (number)                               //按照序号来执行不同模块下的功能
（模块化）
        {
        case 1:
            book[book_num].book_add(book_num);
            break;
        case 2:
            address_book::book_inquire(book_num);
            break;
        case 3:
            address_book::data_display(book_num);
            break;
        case 4:
            address_book::book_delete(book_num);
            break;
        case 5:
            address_book::data_edit(book_num);
            break;
        case 6:
            address_book::book_save(book_num);
            break;
        case 7:
```

```
            address_book::book_read(book_num);
            break;
        case 8:
            confirm_exit();
            break;
        default:
            cout << "请输入 1 到 8 内的数字\n";
            system("pause");
            system("cls");
            break;
        }
    }
```

先判断用户输入的数字数据类型是否正确. 如果不正确则出现错误提示，并返回菜单令其重新输入。如果正确，则进行 switch 判断，按照输入的数字分别调用相关函数，调用完毕后自动返回菜单。system("pause");使程序暂停，方便用户查看显示结果；system("cls"); 是清屏语句，用于清除当前屏幕内容。

3）添加功能

```
void address_book::book_add(int &book_num)                          //添加功能
{
    bool judge = 1;
    do                                                              //如果用
户不停止输入重复编号，则一直要求输入，直到编号不再重复为止。
    {
        cout << "请输入编号：  " << endl;
        while (!(cin >> number))                                    //如果输入的
不符合数据类型要求，就会返回 0. 本句即输入错误时执行以下语句。
        {
            cout << "输入格式有误(应为数字)，请重新输入：  " << endl;
            cin.clear();                                            //与下句同用
方可去掉缓冲区。
            cin.sync();                                             //清除缓冲
区，避免无限循环。
            while (cin.get() != '\n')                               //即输入内容，一
直到遇到回车时停止，再将该语句送回前面判断。
            {
```

```
                continue;                                                    //如果不设
这句，会导致不断读取缓冲区数据，无限循环。
            }
        }
        for (int n = 0; n <= book_num - 1; n++)                              //book[0].number
可忽略不计
        {
            if (book[n].number == number)
            {
                cout << "编号重复！请重新输入：  " << endl;
                break;
            }
            if (n == book_num - 1)                                           //如果一直到
book_num-1 判断完毕后都没有出现重复数字，跳出循环。
            {
                judge = 0;
                break;
            }
        }
    } while (judge);
    while (1)
    {
        cout << "请输入姓名：  " << endl;
        cin >> name;
        cout << "请输入性别：  " << endl;
        cin >> sex;
        cout << "请输入通讯地址：  " << endl;
        cin >> mailing_address;
        cout << "请输入邮箱地址：  " << endl;
        cin >> email_address;
        cout << "请输入电话号码：  " << endl;
        cin >> phone_number;
        if (!book_check(this))                                               //检查函数
        {
            system("pause");
            system("cls");
```

```
            menu_show();
            continue;
        }
        break;
    }
    Pseudo_save("保存");
    book_num++;
    system("pause");
    system("cls");
}
```

先判断用户输入的编号数据类型是否正确。首先要求用户输入编号，如果输入数据类型不符合要求，则 cin 返回一个 0 值，while (!(cin >> number)) 则在用户输入数据类型不符的情况下执行，如果输入的数据类型不匹配，会返回 0 值，因此执行 while 语句，cin.sync()和 cin.clear()并用可以可以改变输入流状态，以避免无限读取缓冲区数据导致程序出错。使用 cin.get()是为了从缓冲区把错误类型的数据全部取出，一直到取出回车符为止，达到清空缓冲区的效果。之后返回 cin>>number 进行重新输入,如果数据类型错误则继续要求重新输入。如果正确则依次录入新增人员的各种信息。之后调用格式正确性检查函数（book_check(this)），数据确认无误后添加入通讯录中。

4）查询功能

```
void address_book::book_inquire(int book_num)
{
    int book_information_int, num; char book_information_str[50];
    bool judge = 0; int n;
    cout << "--------------------------+\n";
    cout << "输入 1 ：按 编号      查找  |" << endl
        << "输入 2 ：按 姓名      查找  |" << endl
        << "输入 3 ：按 电话号码 查找  |" << endl
        << "输入 4：  按 通讯地址 查找  |" << endl
        << "输入 5：  按 邮箱地址 查找  |" << endl;
    cout << "--------------------------+\n";
    cout << "请输入你想查询的方式：        " << endl;
    cin >> num;
    switch (num)
    {
    case 1:
```

```
    cout << "--------------------------+\n";
    cout << "请输入编号：" << endl;
    while (!(cin >> book_information_int))
    {
        cout << "输入格式有误(应为数字)，请重新输入： " << endl;
        cin.clear();
        cin.sync();
        while (cin.get() != '\n')
        {
            continue;
        }
    }
    for (n = 1; n <= book_num; n++)
    {
        if (book[n].number == book_information_int)
        {
            judge = 1;
            break;
        }
    }
    break;
case 2:
    cout << "--------------------------+\n";
    cout << "请输入姓名:" << endl;
    cin >> book_information_str;
    for (n = 1; n <= book_num; n++)
    {
        if (!strcmp(book[n].name,book_information_str))
        {
            judge = 1;
            break;
        }
    }
    break;
case 3:
    cout << "--------------------------+\n";
```

```
    cout << "请输入电话号码:" << endl;
    cin >> book_information_str;
    for (n = 1; n <= book_num; n++)
    {
        if (!strcmp(book[n].phone_number,book_information_str))
        {
            judge = 1;
            break;
        }
    }
    break;
case 4:
    cout << "--------------------------+\n";
    cout << "请输入通讯地址:" << endl;
    cin >> book_information_str;
    for (n = 1; n <= book_num; n++)
    {
        if (!strcmp(book[n].mailing_address,book_information_str))
        {
            judge = 1;
            break;
        }
    }
    break;
case 5:
    cout << "--------------------------+\n";
    cout << "请输入邮箱地址:" << endl;
    cin >> book_information_str;
    for (n = 1; n <= book_num; n++)
    {
        if (!strcmp(book[n].email_address,book_information_str))
        {
            judge = 1;
            break;
        }
    }
```

```
        break;
    }
    if (judge)
    {
        cout << "查找成功！" << endl;
        cout << "--------------------------+\n";
        cout << "编号：" << book[n].number << endl
            << "姓名：  " << book[n].name << endl
            << "性别：  " << book[n].sex << endl
            << "通讯地址：  " << book[n].mailing_address << endl
            << "邮箱地址：" << book[n].email_address << endl
            << "电话号码：  " << book[n].phone_number << endl;
        cout << "--------------------------+\n";
    }
    else
    {
        cout << "查无此人。" << endl;
    }
    system("pause");
    system("cls");
}
```

先询问用户按何种方式进行查找，如果是按照编号方式，用户便输入整型数据（book_information_int），这是因为编号为整型，之后用该数据和通讯录对象数组中的编号一一比对，如有相同则输出该编号所属人员的全部信息。如果是其他方式查找，则令用户输入 char 型数据（book_information_str），因为其他数据的类型为 char 型。对于 char 型的数据，通过 strcmp 比较查找的信息与成员中的某个数据是否完全相同，相同则执行之后的代码。查找成功则输出该成员全部信息，未能查找到则输出“查无此人”。

5）删除功能

```
void address_book::book_delete(int &book_num)
{
    int book_information_int;
    bool judge = 0; int n;
    cout << "--------------------------+\n";
    cout << "请输入要删除记录的编号:" << endl;
    while (!(cin >> book_information_int))
```

```
    {
        cout << "输入格式有误(应为数字)，请重新输入： " << endl;
        cin.clear();
        cin.sync();
        while (cin.get() != '\n')
        {
            continue;
        }
    }
    for (n = 1; n <= book_num; n++)
    {
        if (book[n].number == book_information_int)
        {
            judge = 1;
            break;
        }
    }
    if (judge)
    {
        if (n != book_num)
        {
            while (n != book_num)
            {
                book[n] = book[n + 1];
                n++;
            }
            book_num--;
        }
        else
        {
            book_num--;                    //其实并没有真正删除最后一个元素，只是下标减一，下次添加新元素时会替代原来应该被删除的元素。
        }
        cout << "--------------------------+\n";
        cout << "删除成功！" << endl;
    }
```

```
    else
    {
        cout << "--------------------------+\n";
        cout << "查无此人，无法删除。" << endl;
    }
    system("pause");
    system("cls");
}
```

用户输入编号后进行删除。若成功查询到编号相符合人员，如果该人员所在数组的下标不为最后一个，则使用循环，将该人员下标后一位的对象覆盖该对象，后者不断覆盖前者，一直到倒数第一个覆盖完倒数第二个为止。然后对象数组下标减一，使最后一位对象无法被访问，从而既能够达到删除的效果，也可以避免最后一个对象重复出现。如果该人员所在数组的下标为最后一个，则令对象数组下标减一，使最后一个不可被访问。这样便可以在显示或查询以及编辑通讯录对象时一直不能查看最后一个对象。至于最后一个对象，在下次添加新对象时会覆盖本应删除的这个对象。

6)存储功能

```
void address_book::book_save(int book_num)                          //储存功能
{
    if (book_num == 1)
    {
        cout << "通讯录为空，无法保存。" << endl;
    }
    else
    {
        ofstream outfile("addressbook.txt");                        //保存到
addressbook.txt
        if (!outfile)
        {
            cout << "文件路径错误或文件不存在。" << endl;
        }
        else
        {
            for (int n = 1; n <= book_num - 1; n++)
            {
                outfile.write((char *)&book[n], sizeof(book[n]));          }
```

```
            Pseudo_save("保存");
            outfile.close();
        }
    }
    system("pause");
    system("cls");
}
```

本项目采用二进制形式保存通讯录数据。首先初始化文件流对象 outfile，并提供文件名（addressbook.txt)。保存原理为：将强制转换后的字符指针按顺序依次指向通讯录数组的各个元素，然后将该数组元素所包含的字节数所含数据输出到文本文档（addressbook.txt）中，一直执行到所有元素输出完毕为止。之后使用 outfile.close()中止文件流对象与 addressbook.txt 文件之间的连接。

7）读取功能

```
void address_book::book_read(int &book_num)
{
    int n = 1;
    ifstream infile("addressbook.txt");                                    //保存到addressbook.txt
    if (!infile)
    {
        cout << "文件路径错误或文件不存在。" << endl;
    }
    else
    {
        while (!(infile.eof()))                          //如果没有读取到最后一个数组对象的结尾，继续循环。
        {
            infile.read((char *)&book[n], sizeof(book[n]));        //读取一次指针后移 sizeof(book[n])个字节。
            n++;
        }
        book_num = n - 1;                                 //重新确定 book_num 的个数
        Pseudo_save("读取");
        infile.close();
```

```
    }
    system("pause");
    system("cls");
}
```

该功能是读取外部文件中所有成员信息到内存中储存。先从本地文件中读取通讯录第一个成员，如果没有读取到文件末尾(!infile.eof())，则一直往下读取，一直到全部读取完毕为止。

n 用来确定读取成员的个数，并使其减一后赋给 book_num，目的是为了令下标移动到最后一个成员下标以后的空对象，方便未来添加新成员。因为添加新成员功能是通过读取 book_num 后，直接写入数据到该下标所在的通讯录对象，如果不后移，会导致最后一个元素被新元素覆盖。

此外注意到，当读取到文件末尾后，while 不会立即中止，而是会多循环一次。这是由于 eof 本身性质导致的（即 eof 在发现文件末尾后，并不会立即改变它的状态，而是在再次执行后重新设置状态）。

8）检查功能

```
bool address_book::book_check(address_book *book)   //格式检查
{
    bool name, sex, email_address, flag = true;
    name = sex = true;
    email_address = false;
    char *p = &book->name[0];
    while (*p != '\0')
    {
        if (*p >= '0'&&*p <= '9')
        {
            name = false;
            break;
        }
        p++;
    }
    string p2 = book->sex;                              //由于 1 个汉字占 2 个字节，因此需要转换为 string 类才能识别。
    if (p2 != "男" && p2 != "女")
    {
        sex = false;
```

```
    }
    p = &book->email_address[0];
    while (*p != '\0')
    {
        if (*p == '@')
        {
            email_address = true;
            break;
        }
        p++;
    }
    if (!name)                                          //统计错误的个数，一起输
出。
    {
        cout << "您输入的 姓名 可能不符合格式（应为汉字或英文组合)；" << endl;
        flag = false;
    }
    if (!sex)
    {
        cout << "您输入的 性别 可能不符合格式（应为男或女)；" << endl;
        flag = false;
    }
    if (!email_address)
    {
        cout << "您输入的 电子邮箱 可能不符合格式（应为 xxx@xxx.com)；" << endl;
        flag = false;
    }
    if (!flag)
    {
        cout << "请按 任意键 重新输入:" << endl;
        return false;
    }
    else
    {
        return true;
    }
```

```
}
```

函数中设置指向对象的指针作为形参，接受每个数组对象的地址，设置多个布尔变量对应每个数据的信息。之后对该对象的每个数据成员进行正误检查，经过相应的检查之后。如果信息无误，则设置为 true，有误则设置为 false。又设置一个 flag 变量统一全局，只要其中任意一个信息有误，flag 则设置为 false，之后函数的返回值和 flag 相同，返回值送回其他函数供其使使用。数据全部检查完毕后，统计错误个数，依次输出每项数据错误信息，然后提示用户重新输入，一直到输入格式正确为止。

5 结论与心得体会

本次完成的项目令人收获颇丰。本组所有成员的编程能力和水平皆有所提升，同时也明白了分工合作对于编程的重要性。分工合作能够提高编程的效率，但也需要团队每一个人的配合。成员提出的构想和思路需要通俗易懂，清晰地向其他成员解释，以方便增加他们对程序的理解。关于代码的编写方面，本组成员每次对源代码做出修改后，都会指出更新的内容，并让他人了解本次代码的改动之处和原理。前期的程序编写，出现了变量名不规范，以及 bug 接连不断，或者是程序运行不符合期望的问题，但经过团队的齐心协力，终于攻克了一个又一个的难关。通过网上查阅有关技术点的资料，以及在程龙老师的殷切指导下，解决了不少难题。

众所周知，C++是一门实践性占有重要地位的编程语言，纯看理论知识对于提升自己的编程能力效果有限，只有将编程实践和理论指导相结合，才能使自己对这门语言的了解更加透彻。在本次 C++程序编写中，本人也曾遇到过无法将理论转化为实践的问题，能够理解有关编程知识，却无法用代码表现出来，或者是代码格式不合法,导致无法运行程序。后来通过不断查阅书本，勘正错误，坚持不懈进行代码修改和调试，最后终于能够写出符合规范的代码。在不断的实践中，本人编程水平也得到了较大提升，能力不再局限于书本理论。

本次编写代码的过程中，也出现了原计算机能正常运行的代码在另一台计算机无法正常运行的问题，最后本组发现是由于编译器不同导致的，并非代码问题。这也说明程序出现的错误往往发生于非程序本身之处，对于错误原因的分析需要考虑缜密。

C++是一门功能强大的语言，它能够实现相当多的功能。在本次项目设计中，本小组对其的利用和理解仅仅是冰山一角。但通过本次项目设计，本小组的每位成员对面向对象的编程思想、以及类和对象、函数等概念的理解更加深刻，夯实了编程基础；本小组的每位成员也意识到，目前对 C++语言的理解和掌握的水平还是远远不足的，今后还需对这门语言进行更加深入的学习，掌握更多的编程知识，提升自身的编程水平。

6 参考文献

[1] 谭浩强.C++程序设计（第三版）.北京：清华大学出版社，2015
[2] (美)普拉达.C++ Primer Plus（第六版）.北京：人民邮电出版社，2012

7 附录

7.1 调试报告及问题

调试信息：

“AddressBook.exe”(Win32): 已加载“C:\Windows\SysWOW64\ntdll.dll”。无法查找或打开 PDB 文件。

“AddressBook.exe”(Win32): 已加载“C:\Windows\SysWOW64\kernel32.dll”。无法查找或打开 PDB 文件。

目前存在的问题：

本次项目中本小组曾经尝试设计用户输入到一半后按 ESC 中途退出返回原菜单的功能，并也写出了相关代码。但在运行过程中发现许多问题，例如输入时第一个字符需要输入两次才会显示，以及不输入内容时按 ESC 能够退出返回原菜单，但输入了部分内容后按 ESC 则会清空输入内容，但无法返回原菜单，并且数据在保存时也出了许多问题。在考虑代码稳定性优先、操作对用户友好的原则下，本小组并没有实现该方案。

7.2 测试结果

1）主界面

图 7-2-1 主界面

2）添加功能

输入 1，输入人员有关信息进行添加数据。

图 7-2-2 为用户按照正确格式输入，用户此时输入了名为张世杰的人员的信息，可以注意到，用户输入

的数据都是符合代码中提供的格式要求的（即姓名为汉字或英文组合，性别应为男或女，邮箱地址需要有@.com 相关字样）：

图 7-2-2 正确输入范例

图 7-2-3 为用户按照错误格式输入：

图 7-2-3 错误输入范例

可以注意到，用户输入的姓名为 123，邮箱地址为 abdf，同时性别为 b，这些信息经过检测错误的函数后，会返回一系列错误信息，同时给出用户需要正确输入的格式信息。提醒用户需要进行重新输入。

返回错误提示后，用户按下任意键后，进行重新输入操作。

3）查询功能

用户在主菜单界面键入 2 后按下回车，显示要求用户查找信息的方式，用户再次通过按键进行相关操作，

以下图例为用户输入 2 后的相关操作，即用户执行了按照姓名查找的方式进行查找通讯录成员。图 7-2-4 为用户按照姓名查找：

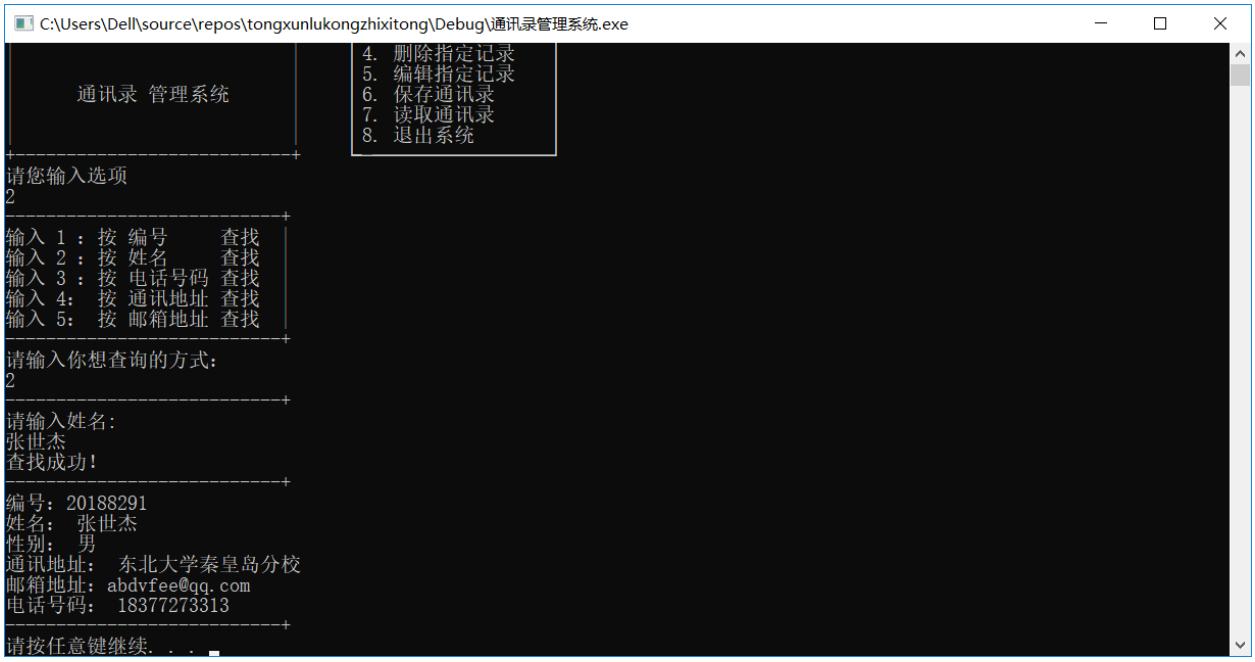


图 7-2-4 按姓名查找

用户再次重复上述操作，之后键入 1（即按编号查找），然后输入他想查询的编号并回车。此时执行相关函数，如果查找到，则输出该人员的所有信息。图 7-2-5 为用户按照编号查找成功的情形：

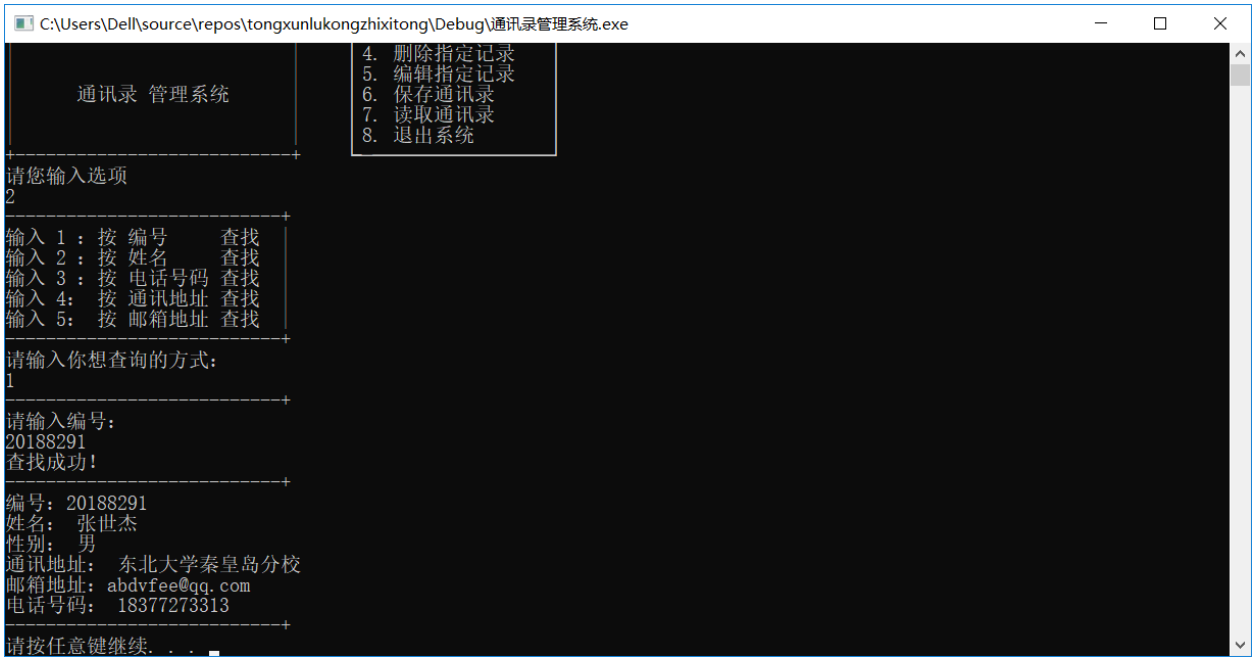


图 7-2-5 按编号查找

如果用户尝试输入通讯录中不存在的编号，便会显示错误信息，图 7-2-6 为用户查找失败的情形：

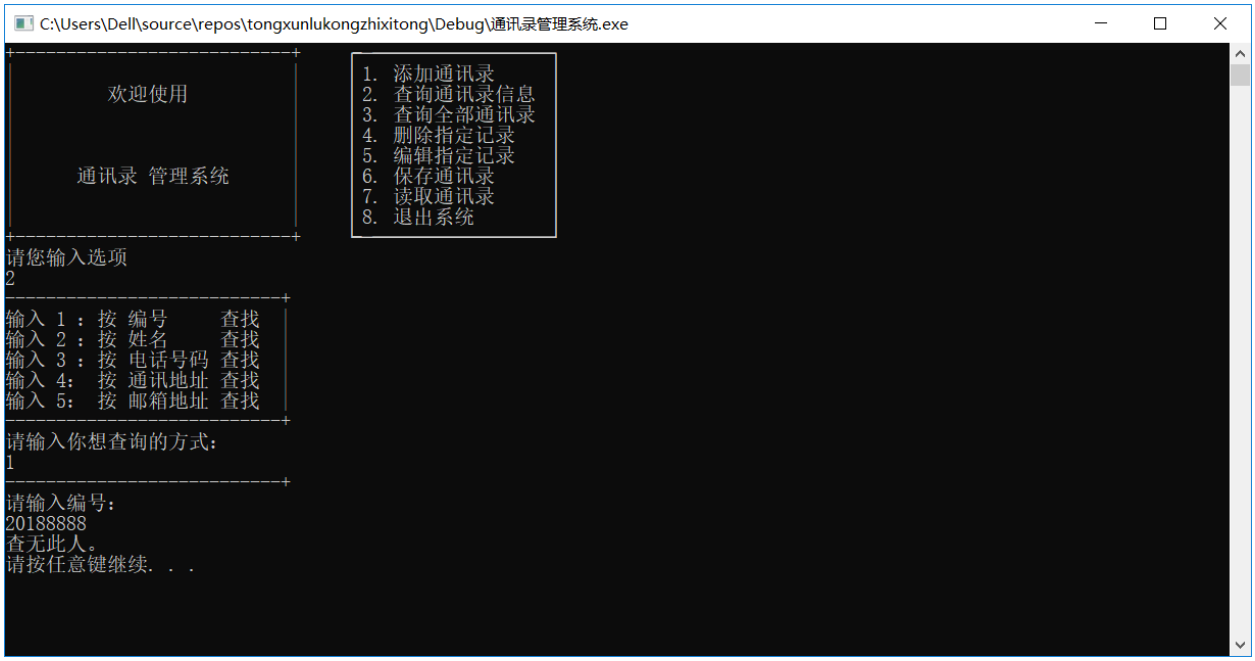


图 7-2-6 查找失败

4）全部查询功能

用户在主菜单界面输入 3 后并按下回车后，便会显示所有人员全部信息，如图 7-2-7 所示：

图 7-2-7 查找全部通讯录

5）删除功能

本例为在 4）的基础上，用户在主菜单界面键入 1，然后输入 20188219，之后执行删除操作。此时用户删除编号为 20188219 的人员，如图 7-2-8 所示：

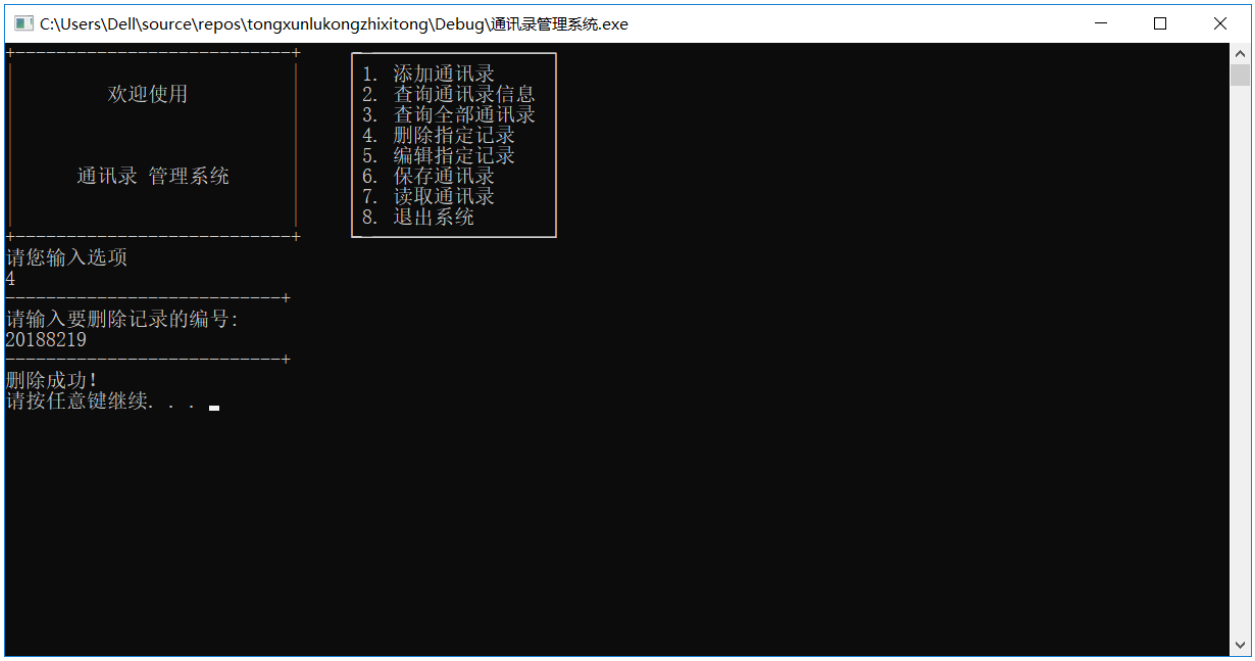


图 7-2-8 删除人员

在用户成功执行删除操作后，用户回到主菜单界面，此时用户再次键入 3 然后按下回车，执行全部通讯录显示功能。此时再次查询全部人员，如图 7-2-9 所示：

图 7-2-9 查询全部人员

可以看到，名单列表中已经没有了用户删除的人员，只显示删除操作过后剩下的人员信息。表明用户完成了删除指定人员的操作。

6）编辑功能

在 5）的基础上，用户在主菜单界面键入 5，执行编辑操作，之后系统会显示要求用户查询的方式并提供相应的数码，用户输入相关数码后按下回车，即进行相关的操作。图例为用户对编号为 20188311 的人员进行修改电话号码。如图 7-2-10 所示：

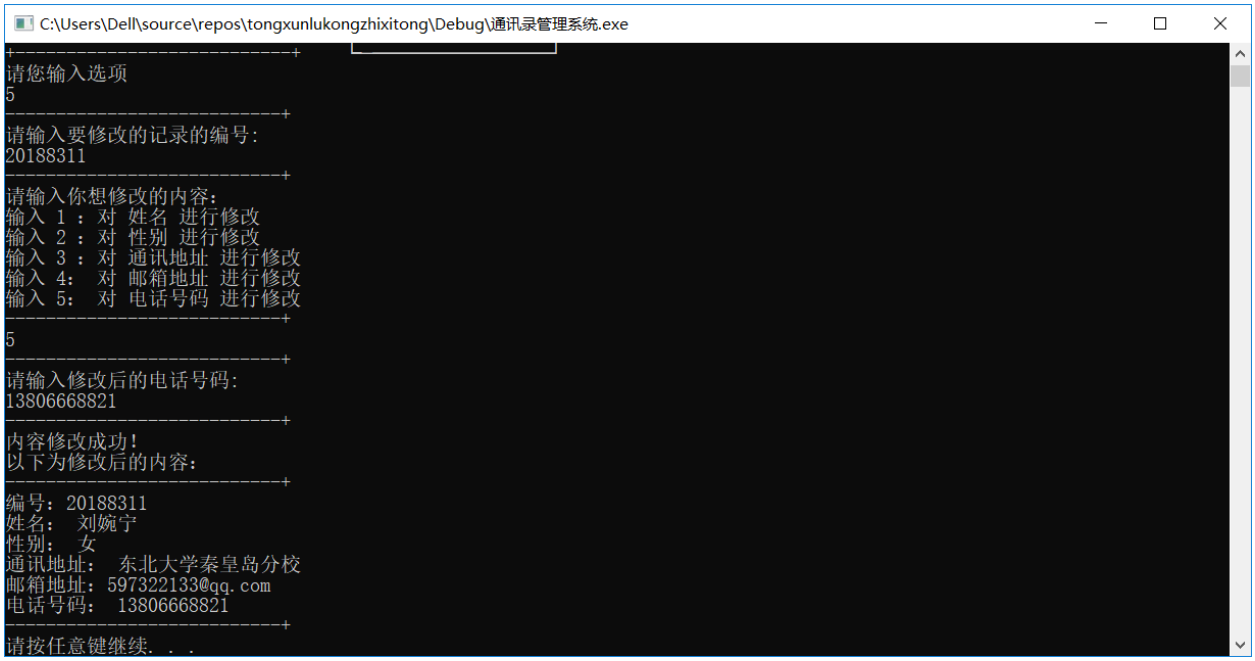


图 7-2-10 修改用户电话号码数据

用户回到主菜单界面后，键入 3 后回车执行全部显示功能，此时查看所有人员信息，如图 7-2-11 所示，指定人员的电话号码已经被修改：

图 7-2-11 修改完毕界面

7）保存和读取功能

用户在主菜单界面输入 6 后按下回车，对已经存入内存的两位成员的相关数据进行保存到外部文件操作，如图 7-2-12 所示：

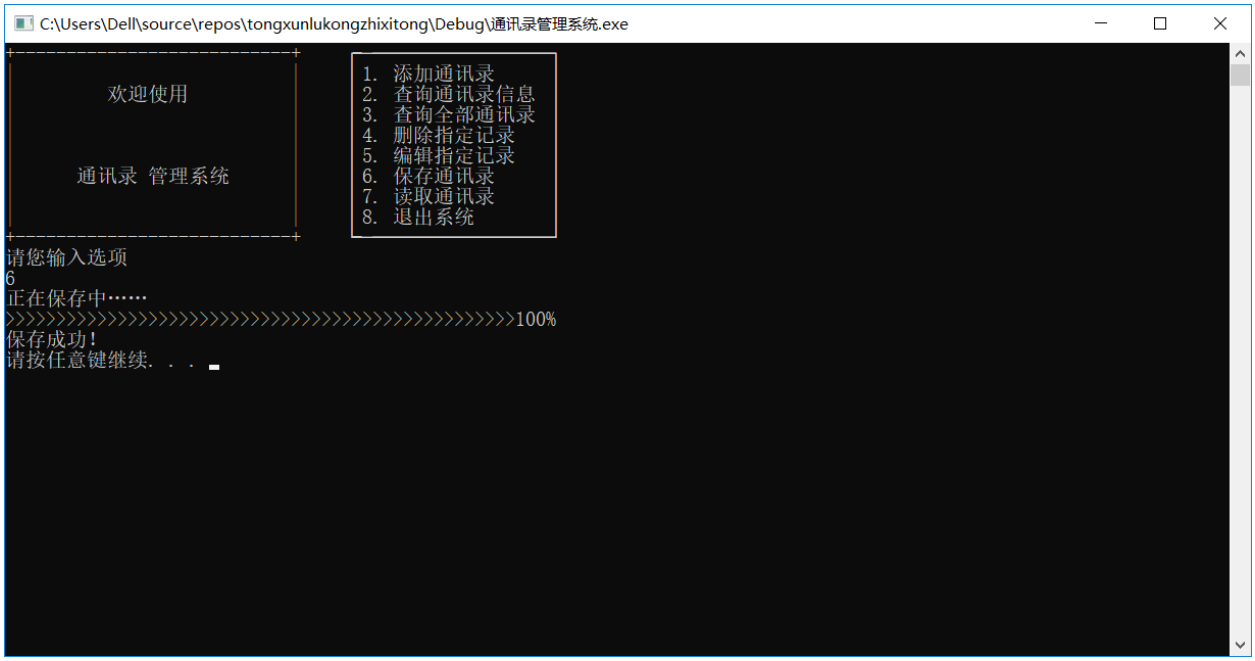
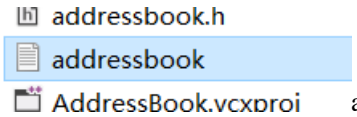


图 7-2-12 保存功能

出现了保存成功提示，表明用户成功执行了保存操作，此时打开该项目文件目录下保存的文件



addressbook.txt，可以查看被保存的数据，如图 7-2-13 所示：

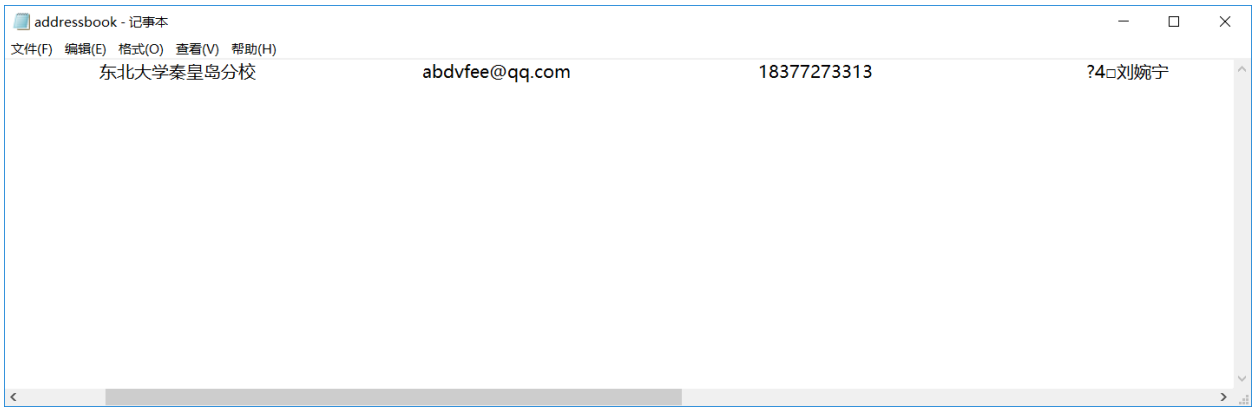


图 7-2-13 addressbook.txt 保存的数据

可见数据已经被成功保存到外部文件当中去，并可以看到之前输入并保存的相关成员的信息。用户回到主菜单界面后，键入 4 并按下回车后，分别输入两个人的编号，然后删除两个人的数据：用户输入 20188291 后并按下回车，删除第一个人员的数据，如图 7-2-1 所示：

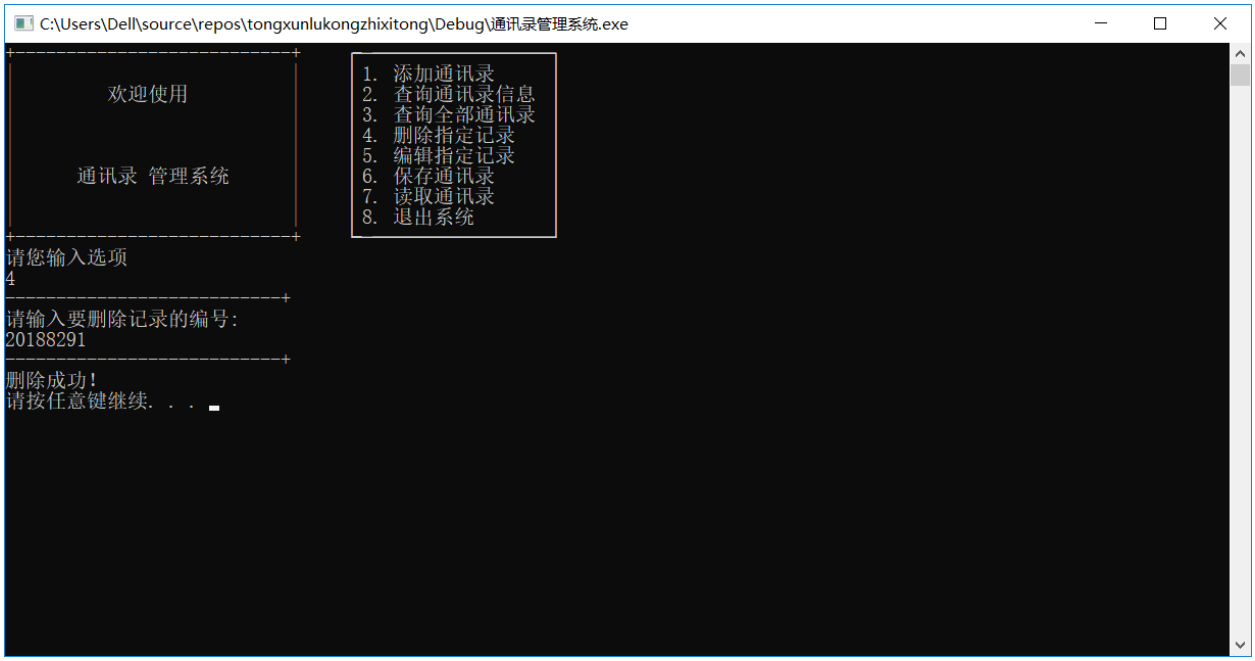


图 7-2-13 删除操作（一）

出现删除成功提示，表明用户删除成功，此时用户输入 20188311 并按下回车，删除第二个人员的数据，如图 7-2-14 所示：

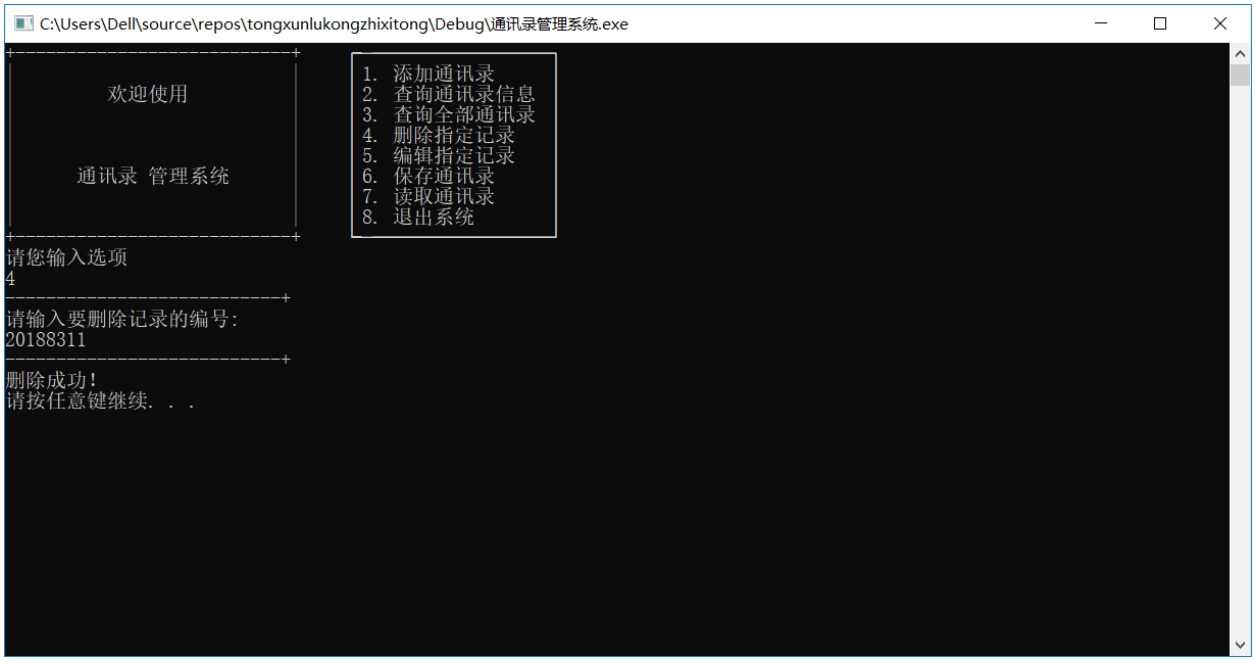


图 7-2-14 删除操作（二）

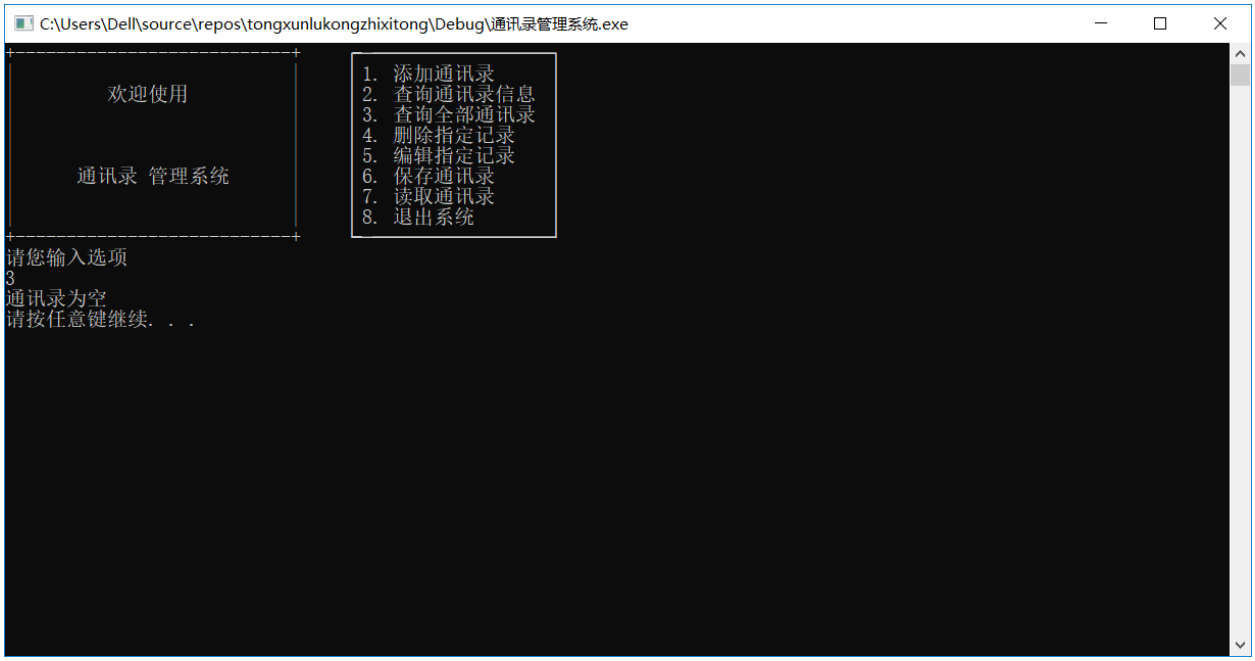


图 7-2-15 删除操作（三）

删除完毕后，用户回到主菜单，键入 3 并按下回车后，执行全部显示的功能。可以看到，通讯录为空，说明所有对象都被删除完毕。这时用户按下 7 键，进行读取功能的操作。将保存在 addressbook.txt 的成员信息全部依次读入进内存当中，如图 7-2-16 所示：

图 7-2-16 读取功能

出现读取成功提示，表明用户成功进行了删除操作，此时用户回到主菜单界面后，键入 3 后按下回车，执行全部查询操作，再次查看全部人员信息，如图 7-2-17 所示：

图 7-2-17 查询人员信息

可以看到，两位之前被成功保存的成员又再次返回到了内存中，这是因为原本的成员被保存到了外部文件，导致进行读取操作时他们的数据得到恢复。

8）退出系统功能

用户按下 8 并回车后，会输出相关文字，要求用户确认是否真正需要退出系统，如果用户按下 Y 就执行关闭程序的操作，如果按下 N 就返回到主菜单，如图 7-2-18 所示：

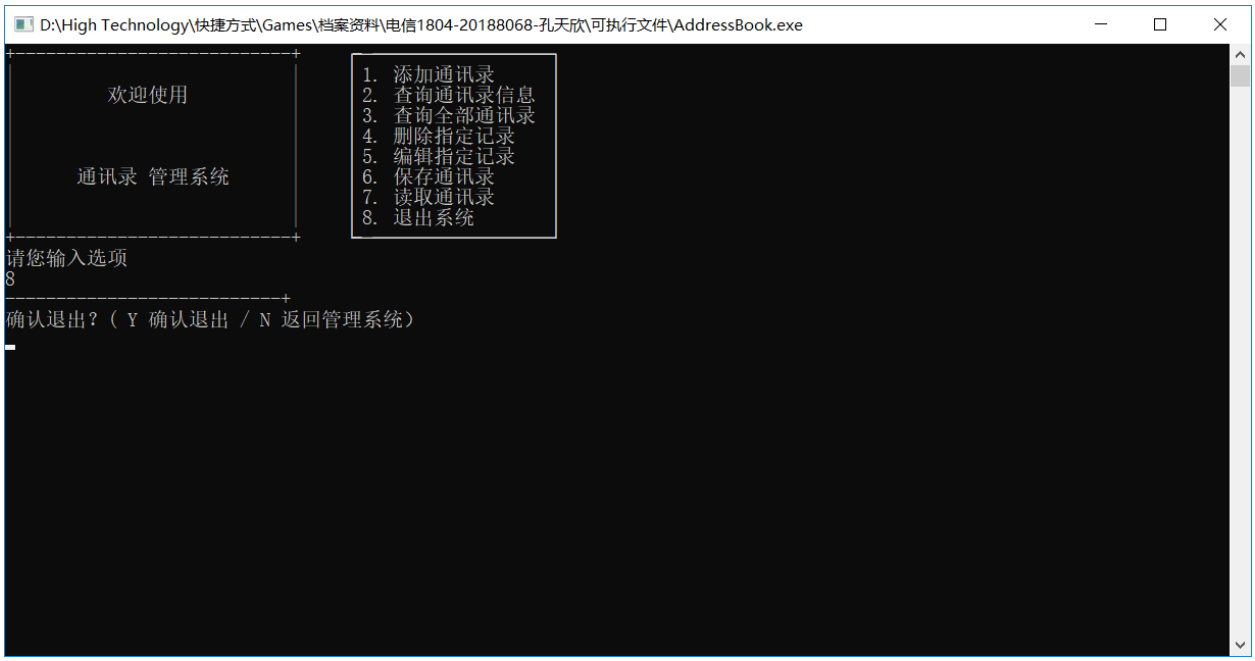


图 7-2-18 退出系统确认界面

用户按下 Y 并回车后，程序会自行关闭。

按下 N 并回车后，返回管理系统。

7.3 关键源代码

//该源代码使用 VS2017 能够正常运行，使用 DevC++和 CFree5.0 运行以下代码存在较小报错的可能性。

//addressbook.cpp 文件部分

```
#include"pch.h"                                    //使用其他编译器时删除本句。
#include <iostream>
#include <string>
#include<iomanip>
#include<fstream>
#include<windows.h>
#include"addressbook.h"
using namespace std;


void menu_show()                                                                    //菜单展示
{

    cout << "+--------------------------+      ┌-──────────────────────────────────┐ \n";
    cout << "|                          |      |  1. 添加通讯录       | \n";
    cout << "|          欢迎使用        |      |  2. 查询通讯录信息   | \n";
    cout << "|                          |      |  3. 查询全部通讯录   | \n";
    cout << "|                          |      |  4. 删除指定记录     | \n";
    cout << "|                          |      |  5. 编辑指定记录     | \n";
    cout << "|      通讯录 管理系统     |      |  6. 保存通讯录       | \n";
    cout << "|                          |      |  7. 读取通讯录       | \n";
    cout << "|                          |      |  8. 退出系统         | \n";
    cout << "+--------------------------+      └-──────────────────────────────────┘ \n";
    cout << "请您输入选项" << endl;
}

bool address_book::book_check(address_book *book)     //格式检查
{
    bool name, sex, email_address, flag = true;
    name = sex = true;
    email_address = false;
    char *p = &book->name[0];
    while (*p != '\0')
    {
        if (*p >= '0'&&*p <= '9')
        {
            name = false;
```

```
            break;
        }
        p++;
    }
    string p2 = book->sex;                          //由于 1 个汉字占 2 个字节，因此需要转换为
string 类才能识别。
    if (p2 != "男" && p2 != "女")
    {
        sex = false;
    }
    p = &book->email_address[0];
    while (*p != '\0')
    {
        if (*p == '@')
        {
            email_address = true;
            break;
        }
        p++;
    }
    if (!name)                                      //统计错误的个数，一起输出。
    {
        cout << "您输入的 姓名 可能不符合格式（应为汉字或英文组合）；" << endl;
        flag = false;
    }
    if (!sex)
    {
        cout << "您输入的 性别 可能不符合格式（应为男或女）；" << endl;
        flag = false;
    }
    if (!email_address)
    {
        cout << "您输入的 电子邮箱 可能不符合格式（应为 xxx@xxx.com）；" << endl;
        flag = false;
    }
    if (!flag)
    {
        cout << "请按 任意键 重新输入:" << endl;
        return false;
    }
    else
    {
        return true;
    }
```

```
}
void Pseudo_save(string sl)                                    //进度条动画
{
    int a, b;
    cout << "正在" << sl << "中……";
    cout << endl;
    for (b = 1; b <= 53; b++)
    {
        cout << "=";
    }
    for (a = 1; a <= 50; a++)
    {
        cout << "\r";                                          //转至本行首位
        for (b = 1; b <= a; b++)
        {
            cout << ">";
        }
        cout.width(3);                                         //设置输出宽度 3
        cout << 2 * a << "%";                                  //百分比号
        Sleep(35);                                             //设置时间
    }
    cout << endl;
    cout << sl << "成功！" << endl;
}

void address_book::book_save(int book_num)                     //储存功能
{
    if (book_num == 1)
    {
        cout << "通讯录为空，无法保存。" << endl;
    }
    else
    {
        ofstream outfile("addressbook.txt");                   //保存到 addressbook.txt
        if (!outfile)
        {
            cout << "文件路径错误或文件不存在。" << endl;
        }
        else
        {
            for (int n = 1; n <= book_num - 1; n++)
            {
                outfile.write((char *)&book[n], sizeof(book[n]));    //保存一次指针后移 sizeof(book[n])
个字节。
            }
```

```
            Pseudo_save("保存");
            outfile.close();
        }
    }
    system("pause");
    system("cls");
}

void address_book::book_read(int &book_num)
{
    int n = 1;
    ifstream infile("addressbook.txt");                    //保存到 addressbook.txt
    if (!infile)
    {
        cout << "文件路径错误或文件不存在。" << endl;
    }
    else
    {
        while (!(infile.eof()))
        {
            infile.read((char *)&book[n], sizeof(book[n]));
            n++;
        }
        book_num = n-1;                                    //重新确定 book_num 的个数，并使下标移至最
后一个元素以后
        Pseudo_save("读取");
        infile.close();

    }
    system("pause");
    system("cls");
}


void address_book::book_add(int &book_num)                 //添加功能
{
    bool judge = 1;
    do                                                     //如果用户不停止输
入重复编号，则一直要求输入，直到编号不再重复为止。
    {
        cout << "请输入编号：  " << endl;
        while (!(cin >> number))                           //如果输入的不符合数据
类型要求，就会返回 0. 本句即输入错误时执行以下语句。
        {
```

```
                cout << "输入格式有误(应为数字），请重新输入：  " << endl;
                cin.clear();                                                    //与下句同用方可去掉缓
冲区。
                cin.sync();                                                     //清除缓冲区，避免无限
循环。
                while (cin.get() != '\n')                                       //即输入内容，一直到遇到回
车时停止，再将该语句送回前面判断。
                {
                    continue;                                                   //如果不设这句，会导
致不断读取缓冲区数据，无限循环。
                }
            }
            for (int n = 0; n <= book_num - 1; n++)                             //book[0].number 可忽略不计
            {
                if (book[n].number == number)
                {
                    cout << "编号重复！请重新输入：  " << endl;
                    break;
                }
                if (n == book_num - 1)                                          //如果一直到
book_num-1 判断完毕后都没有出现重复数字，跳出循环。
                {
                    judge = 0;
                    break;
                }
            }
        } while (judge);
        while (1)
        {
            cout << "请输入姓名：  " << endl;
            cin >> name;
            cout << "请输入性别：  " << endl;
            cin >> sex;
            cout << "请输入通讯地址：  " << endl;
            cin >> mailing_address;
            cout << "请输入邮箱地址：  " << endl;
            cin >> email_address;
            cout << "请输入电话号码：  " << endl;
            cin >> phone_number;
            if (!book_check(this))                                              //检查函数
            {
                system("pause");
                system("cls");
                menu_show();
                continue;
```

```
            }
            break;
        }
        Pseudo_save("保存");
        book_num++;
        system("pause");
        system("cls");
}

void address_book::book_inquire(int book_num)
{
        int book_information_int, num; char book_information_str[50];
        bool judge = 0; int n;
        cout << "--------------------------+\n";
        cout << "输入 1 ：按 编号        查找   |" << endl
             << "输入 2 ：按 姓名        查找   |" << endl
             << "输入 3 ：按 电话号码  查找   |" << endl
             << "输入 4：  按 通讯地址  查找   |" << endl
             << "输入 5：  按 邮箱地址  查找   |" << endl;
        cout << "--------------------------+\n";
        cout << "请输入你想查询的方式：            " << endl;
        cin >> num;
        switch (num)
        {
        case 1:
            cout << "--------------------------+\n";
            cout << "请输入编号：" << endl;
            while (!(cin >> book_information_int))
            {
                cout << "输入格式有误(应为数字），请重新输入：  " << endl;
                cin.clear();
                cin.sync();
                while (cin.get() != '\n')
                {
                    continue;
                }
            }
            for (n = 1; n <= book_num; n++)
            {
                if (book[n].number == book_information_int)
                {
                    judge = 1;
                    break;
                }
            }
```

```
        break;
    case 2:
        cout << "----------------------------+\n";
        cout << "请输入姓名:" << endl;
        cin >> book_information_str;
        for (n = 1; n <= book_num; n++)
        {
            if (!strcmp(book[n].name,book_information_str))
            {
                judge = 1;
                break;
            }
        }
        break;
    case 3:
        cout << "----------------------------+\n";
        cout << "请输入电话号码:" << endl;
        cin >> book_information_str;
        for (n = 1; n <= book_num; n++)
        {
            if (!strcmp(book[n].phone_number,book_information_str))
            {
                judge = 1;
                break;
            }
        }
        break;
    case 4:
        cout << "----------------------------+\n";
        cout << "请输入通讯地址:" << endl;
        cin >> book_information_str;
        for (n = 1; n <= book_num; n++)
        {
            if (!strcmp(book[n].mailing_address,book_information_str))
            {
                judge = 1;
                break;
            }
        }
        break;
    case 5:
        cout << "----------------------------+\n";
        cout << "请输入邮箱地址:" << endl;
        cin >> book_information_str;
        for (n = 1; n <= book_num; n++)
```

```
            {
                if (!strcmp(book[n].email_address,book_information_str))
                {
                    judge = 1;
                    break;
                }
            }
            break;
        }
        if (judge)
        {
            cout << "查找成功！" << endl;
            cout << "--------------------------+\n";
            cout << "编号：" << book[n].number << endl
                << "姓名：  " << book[n].name << endl
                << "性别：  " << book[n].sex << endl
                << "通讯地址：  " << book[n].mailing_address << endl
                << "邮箱地址：" << book[n].email_address << endl
                << "电话号码：  " << book[n].phone_number << endl;
            cout << "--------------------------+\n";
        }
        else
        {
            cout << "查无此人。" << endl;
        }
        system("pause");
        system("cls");
}

void address_book::data_display(int book_num)
{
        if (book_num == 1)                    //没有添加过任何通讯录信息的 book_num 为 1.
        {
            cout << "通讯录为空" << endl;

        }
        else
        {
            cout << std::left << "  编号  " << std::right << setw(7) << "姓名" << std::right << setw(8) << "性别"
<< std::right << setw(17)
                << "通讯地址" << std::right << setw(21) << "邮箱地址" << std::right << setw(19) << "电话号
码" << endl;
            cout << "-------------------------------------------------------------------------------------------" << endl;
            for (int n = 1; n <= book_num - 1; n++)
            {
```

```
                cout << std::left
                    << book[n].number << std::right << setw(8)
                    << book[n].name << std::right << setw(6)
                    << book[n].sex << std::right << setw(24)
                    << book[n].mailing_address << std::right << setw(20)
                    << book[n].email_address << std::right << setw(16)
                    << book[n].phone_number << endl;
            }
            cout << "-----------------------------------------------------------------------------------------------" << endl;
        }
        system("pause");
        system("cls");
}

void address_book::book_delete(int &book_num)
{
        int book_information_int;
        bool judge = 0; int n;
        cout << "---------------------------+\n";
        cout << "请输入要删除记录的编号:" << endl;
        while (!(cin >> book_information_int))
        {
            cout << "输入格式有误(应为数字），请重新输入：  " << endl;
            cin.clear();
            cin.sync();
            while (cin.get() != '\n')
            {
                continue;
            }
        }
        for (n = 1; n <= book_num; n++)
        {
            if (book[n].number == book_information_int)
            {
                judge = 1;
                break;
            }
        }
        if (judge)
        {
            if (n != book_num)
            {
                while (n != book_num)
                {
                    book[n] = book[n + 1];
```

```
                n++;
            }
            book_num--;
        }
        else
        {
            book_num--;
//其实并没有真正删除最后一个元素，只是下标减一，下次添加新元素时会替代原来应该被删除的元素。
        }
        cout << "--------------------------+\n";
        cout << "删除成功！" << endl;
    }
    else
    {
        cout << "--------------------------+\n";
        cout << "查无此人，无法删除。" << endl;
    }
    system("pause");
    system("cls");
}


void address_book::data_edit(int &book_num)
{
    char book_information_str[50]; int book_information_int;
    bool judge = 0; int n, num;
    cout << "--------------------------+\n";
    cout << "请输入要修改的记录的编号:" << endl;
    while (!(cin >> book_information_int))
    {
        cout << "输入格式有误(应为数字），请重新输入：  " << endl;
        cin.clear();
        cin.sync();
        while (cin.get() != '\n')
        {
            continue;
        }
    }
    for (n = 1; n <= book_num; n++)
    {
        if (book[n].number == book_information_int)
        {
            judge = 1;
            break;
        }
```

```
}
if (judge)
{
    cout << "---------------------------+\n";
    cout << "请输入你想修改的内容：  " << endl;
    cout << "输入 1 ：对 姓名 进行修改" << endl
        << "输入 2 ：对 性别 进行修改" << endl
        << "输入 3 ：对 通讯地址 进行修改" << endl
        << "输入 4：  对 邮箱地址 进行修改" << endl
        << "输入 5：  对 电话号码 进行修改" << endl;
    cout << "---------------------------+\n";
    cin >> num;
    switch (num)
    {
    case 1:
        cout << "---------------------------+\n";
        cout << "请输入修改后的姓名：" << endl;
        cin >> book_information_str;
        strcpy_s(book[n].name, book_information_str);
        judge = 1;
        break;
    case 2:
        cout << "---------------------------+\n";
        cout << "请输入修改后的性别:" << endl;
        cin >> book_information_str;
        strcpy_s(book[n].sex, book_information_str);
        judge = 1;
        break;
    case 3:
        cout << "---------------------------+\n";
        cout << "请输入修改后的通讯地址:" << endl;
        cin >> book_information_str;
        strcpy_s(book[n].mailing_address, book_information_str);
        judge = 1;
        break;
    case 4:
        cout << "---------------------------+\n";
        cout << "请输入修改后的邮箱地址:" << endl;
        cin >> book_information_str;
        strcpy_s(book[n].email_address, book_information_str);
        judge = 1;
        break;
    case 5:
        cout << "---------------------------+\n";
```

```
            cout << "请输入修改后的电话号码:" << endl;
            cin >> book_information_str;
            strcpy_s(book[n].phone_number, book_information_str);
            judge = 1;
            break;

        }
        cout << "--------------------------+\n";
        cout << "内容修改成功！" << endl;
        cout << "以下为修改后的内容：" << endl;
        if (judge)
        {
            cout << "--------------------------+\n";
            cout << "编号：" << book[n].number << endl
                 << "姓名：  " << book[n].name << endl
                 << "性别：  " << book[n].sex << endl
                 << "通讯地址：  " << book[n].mailing_address << endl
                 << "邮箱地址：" << book[n].email_address << endl
                 << "电话号码：  " << book[n].phone_number << endl;
            cout << "--------------------------+\n";
        }
    }
    else
    {
        cout << "--------------------------+\n";
        cout << "查无此人，无法编辑。" << endl;
    }
    system("pause");
    system("cls");
}
void confirm_exit()
{
    char judge;
    cout << "--------------------------+\n";
    cout << "确认退出？(Y 确认退出 / N 返回管理系统）" << endl;
    cin >> judge;
    if (judge == 'Y' || judge == 'y')
    {
        exit(0);
    }
    else
    {
        system("cls");
    }
```

```
}

int main()
{
    int book_num = 1;                                   //通讯录 数组,从 1 开始.
    int number;                                         //输入的号码
    while (1)                                           //反复循环，来重复输入。
    {
        menu_show();
        while (!(cin >> number))                        //输入错误号码
        {
            cout << "请输入 1 到 8 内的数字" << endl;
            cin.clear();
            cin.sync();
            while (cin.get() != '\n')
            {
                continue;
            }
        }                                               //已经输入正确号码
        switch (number)                                 //按照序号来执行不同模块下的功能（模块化）
        {
        case 1:
            book[book_num].book_add(book_num);
            break;
        case 2:
            address_book::book_inquire(book_num);
            break;
        case 3:
            address_book::data_display(book_num);
            break;
        case 4:
            address_book::book_delete(book_num);
            break;
        case 5:
            address_book::data_edit(book_num);
            break;
        case 6:
            address_book::book_save(book_num);
            break;
        case 7:
            address_book::book_read(book_num);
            break;
        case 8:
            confirm_exit();
            break;
```

```
            default:
                cout << "请输入 1 到 8 内的数字\n";
                system("pause");
                system("cls");
                break;
            }
        }
}

//addressbook.h 头文件部分

#include"pch.h"                                          //使用其他编译器时删除本句。


class address_book                                       //定义通讯录类
{
private:
    int number;                                          //编号
    char name[20];                                       //姓名
    char sex[20];                                        //性别
    char mailing_address[50];                            //通讯地址
    char email_address[50];                              //邮箱地址
    char phone_number[50];                               //电话号码
public:
    void book_add(int &book_num);                        //添加功能
    static void data_display(int book_num);              //全部展示功能
    static void book_inquire(int book_num);              //查询功能
    static void book_delete(int &book_num);              //删除功能
    static void data_edit(int &book_num);                //编辑功能
    static void book_save(int book_num);                 //保存功能
    static void book_read(int &book_num);                //读取功能
    bool book_check(address_book *book);                 //格式检查
}book[100];
```