沈阳工业大学

计算机程序设计实践

课程设计报告

班　　级　计算机 1603 班

学　　号　160405319

姓　　名　张升旺

指导教师　于霞　岳笑含

日　　期　2018.1.5

成　　绩

ThreeBand 智能手环程序设计

1 引言

智能手环是一种穿戴式智能设备。通过这款手环，用户可以记录日常生活中的锻炼、睡眠、部分还有饮食等实时数据，并将这些数据与手机、平板同步，起到通过数据指导健康生活的作用。

智能手环这种设计风格对于习惯佩戴首饰的用户而言，颇具诱惑力。更重要的是，手环的设计风格堪称白搭。而且，别看小小手环个头不大，其功能还是比较强大的，比如它可以说是一款高档的计步器，具有普通计步器的一般计步，测量距离、卡路里、脂肪等功能。同时还具有睡眠监测、高档防水、蓝牙 4.0 数据传输、疲劳提醒的特殊功能。

本课程设计中，我们通过程序设计来简单的进行对智能手环的程序的模拟，以实现基础的智能手环的功能，加强程序设计的能力，以及做一个小项目的团队合作能力。

2 需求分析

2.1 系统功能分析

（1） 时间显示
（2） 智能计步
（3） 运动监测
（4） 来电提醒
（5） 查看、编辑和删除通讯录
（6） 打电话：显示通讯录信息、有电话拨通的声音

2.2 系统设计目标

（1） 实现功能分析中基本的功能需求
（2） 实现通讯录数据通过文件操作进行保存
（3） 模拟实际情况中的操作，增添“SmartPhone”类，实现手环与手机的数据互通
（4） 通过键盘接收数据来模拟计步与心率的计算
（5） 增添“PerInfo”类，存储用户个人信息，并利用文件操作保存用户信息
（6） 可视化界面操作，声音播放与操作可同时进行
（7） 计步与心率测试的模拟情况更真实

3 系统设计

3.1 系统功能设计

本系统共设计五个基础模块测心率、计步、健康信息显示、拨打电话、通讯录

1. 测心率与计步模块均由 ThreeBand 中的一个成员函数实现其功能

2. 健康信息显示模块由 ThreeBand 中 PerInfo 对象成员来记录个人健康数据，并计算出 BMI 值。结合实际情况，采用了文件操作，对个人健康数据进行了保存，来避免程序下一次执行过程中数据的丢失

3. 拨打电话功能模块根据手环与手机数据交换的实际情况，由手环输入电话号码并调用 SmartPhone 中打电话这个函数来具体模拟打电话这个功能

4. 通讯录这个功能模块由于实际使用过程中的需要，额外抽象了联系人这个类，并利用文件操作保存数据，且提供了编辑、添加、删除、打电话的功能。

3.2 类设计与类结构

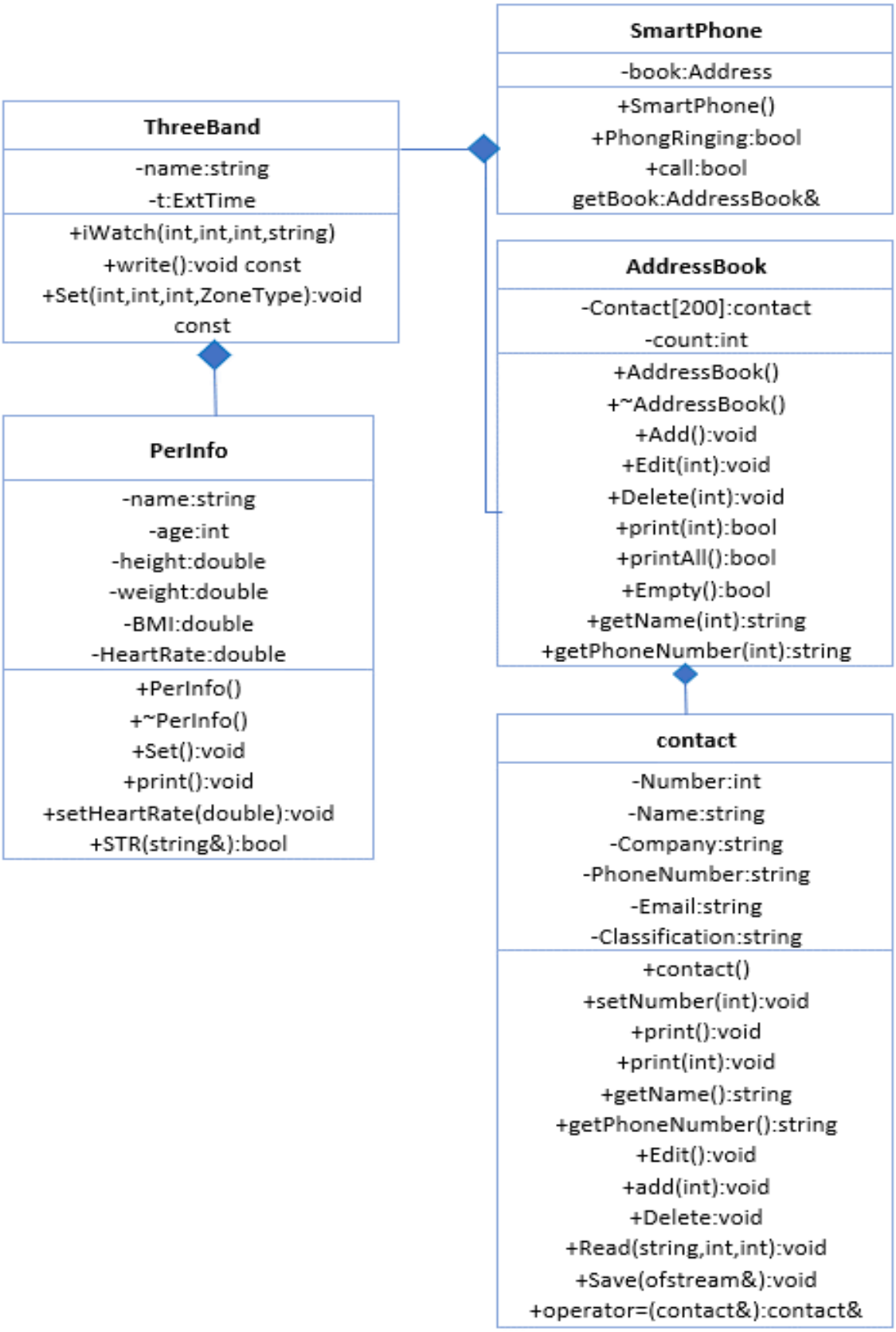

图 1　ThreeBand 程序设计 UML 图

3.3 开发与运行环境

表 1　运行设备详情

项目	型号
电脑型号	惠普暗影精灵Ⅱ OMEN 15-ax016TX
操作系统	Windows 10家庭版
处理器	Intel 6代 酷睿 i5-6300HQ
内存	8G DDR4
显卡	NVIDIA GeForce GTX 960M
编译环境	Microsoft Visual Studio Community 2017
运行环境	Win10、Win8、Win7

4　设计与实现

4.1　功能模块的设计

通讯录模块由通讯录类 AddressBook、联系人类 Contact 以 Composition 方式来实现其添加、删除、修改的基本功能，并结合实际应用情况，利用文件#include<fstream>头文件，以及函数 ifstream、ofstream 来实现对联系人信息的读取以及存储。

该模块的实现难点主要包括一下两个部分：

1. 联系人信息从文件中读取的过程，即从文件中读取的长字符串如何将数据分割，并分别赋予不同的联系人不同的信息。
2. 以及功能需求中的“编号”这个联系人属性，如何在添加和删除的工程中自动修改数值。
3. 删除联系人的具体功能实现。

针对难点 1

我们将联系人数据的读取和储存功能的实现分别放置与 AddressBook 类的构造函数以及析构函数当中。用“{}”来区分每个联系人，用“*”区分联系人的不同信息。如图所示

针对难点 2

我们将编号这一数据成员使用 setNumber 函数来进行自动赋值，在进行每一次增加联系人，或者删除联系人的时候相应修改联系人的编号。而不会导致删除联系人后，下一联系人的编号未发生改变。

针对难点 3

在设计算法过程当中，由于删除的特殊性，而不能仅仅将被删除联系人的信息置“NULL”，还需要使该联系人的存储空间可由之后的新建联系人存储信息。故我们不仅将联系人信息清空，同时将该联系人通过交换，置与已保存联系人之后，以达到再利用的目的。

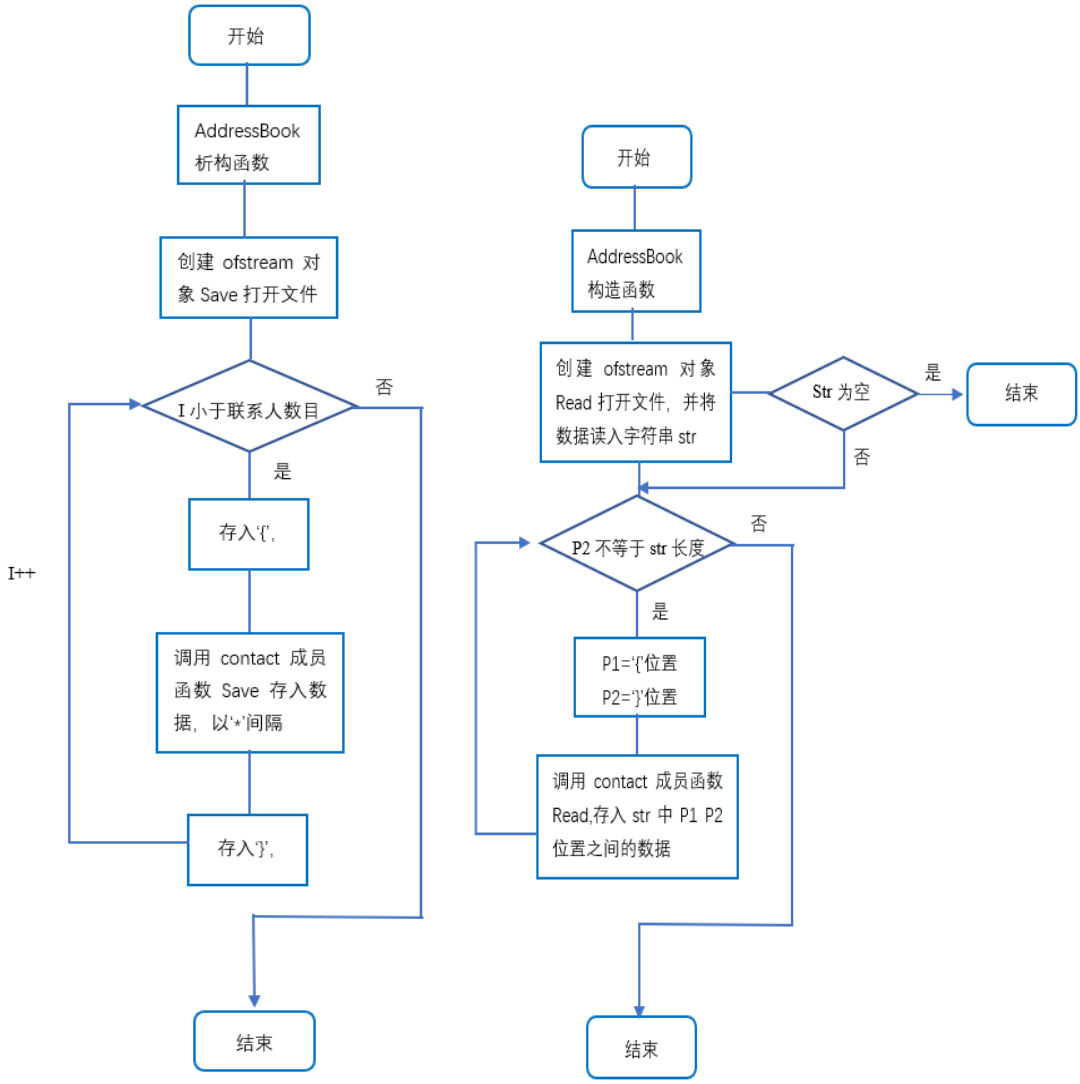

图 2　AddressBook 的联系人数据的读取与保存

4.2 测试

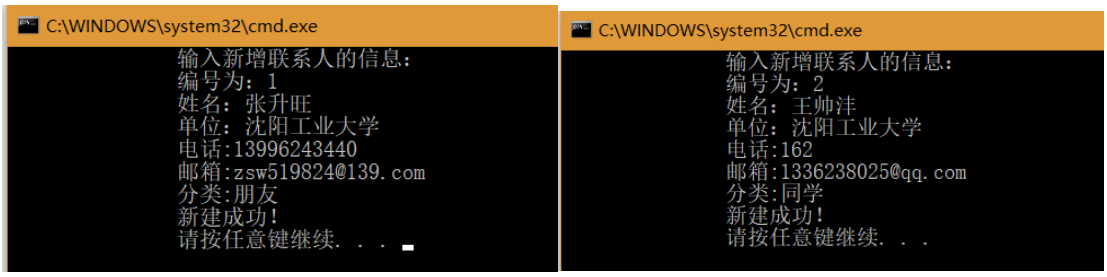

图 3　新建联系人

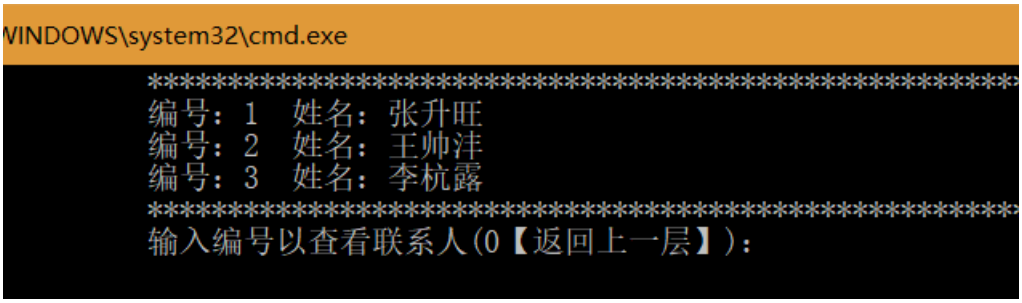

图 4　关闭程序后再打开，显示所有联系人

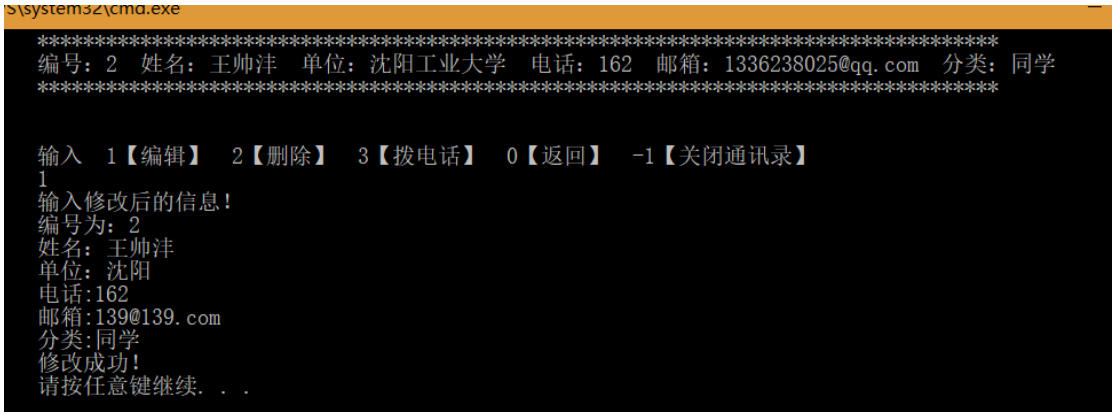

图 5　修改联系人信息

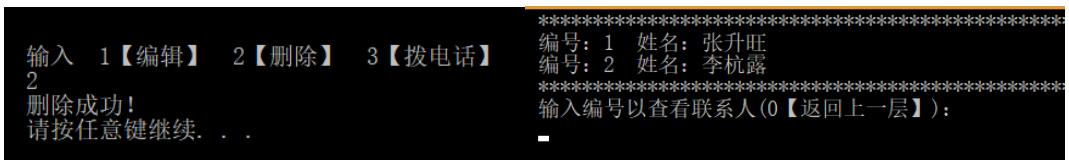

图 6　删除联系人

5 结语

5.1 结论与讨论

该系统达到了以下目标

（1） 实现功能分析中基本的功能需求
（2） 实现通讯录数据通过文件操作进行保存
（3） 模拟实际情况中的操作，增添“SmartPhone”类，实现手环与手机的数据互通
（4） 通过键盘接收数据来模拟计步与心率的计算
（5） 增添“PerInfo”类，存储用户个人信息，并利用文件操作保存用户信息

并实现了项目所需的所有功能

在程序设计过程当中，由于对利用文件存储数据的函数使用不理解，对播放声音的功能实现不了解。我们查阅了大量的网上相关的应用教程，学习了各种函数的使用方法，并在程序中加以实现和利用。由于考虑不够周全，没有做好异常捕获，导致程序经常崩溃，以及出错，我们及时增加了错误操作的处理以及提示，来保障程序的健壮性，以及完整性。

5.2 设计体会

在本课程设计当中，通过对智能手环这个项目的分析、需求分析、类抽象、函数抽象、以及到最终的代码实现程序调试的过程当中，我加深了对本学期 C++学习以来的一些模糊知识、概念的理解，并充分体会到了在程序设计中程序设计思想、实际问题转化为计算机问题这中思想的重要性。

并在之后的程序调试以及测试过程当中，由于考虑不周全，导致用户在输入数据的过程当中产生了程序崩溃 出错的问题。明白了程序健壮性的重要性，在今后的程序设计过程当中也会注意这个问题，注意程序设计的严谨性。

《计算机程序设计实践》课程设计任务划分

姓 名	设计职务	任 务	工作量比例
张升旺	组长	结构设计 算法设计	40%
李杭露	组员	程序测试 项目需求函数资料的查找 绘制 UML 图	30%
王帅沣	组员	项目需求分析 程序调试以及漏洞修复	30%

附录

//File：head.h

```
#include<iostream>
#include<string>
#include<fstream>
#include <cstdlib>
#include <time.h>
#include <conio.h>
class contact
{
public:
    contact()
    {
        Name = "null";
        Company = "null";
        PhoneNumber = "null";
        Email = "null";
        Classification = "null";
    }
    void setNumber(int n) { Number = n; }
    void print();
    void print(int);
    string getName() { return Name; }
    string getPhoneNumber() { return PhoneNumber; }
    void Edit();
    void add(int);
    void Delete();
    void Read(string, int, int);
    void Save(ofstream&);
    contact& operator=(contact&);
private:
    int Number;
    string Name;
    string Company;
    string PhoneNumber;
    string Email;
    string Classification;
};
```

```
class AddressBook
{
public:
    AddressBook();
    ~AddressBook();
    void Add();
    void Edit(int);
    void Delete(int);
    bool print(int);
    bool printAll();
    bool empty() { return count; }
    string getName(int i) { return Contact[i - 1].getName(); }
    string getPhoneNumber(int i) { return Contact[i -
1].getPhoneNumber(); }
private:
    friend class SmartPhone;
    contact Contact[200];
    int count;
};
```

//File：AddressBook_definition.cpp

```
#include"head.h"

BOOL MByteToWChar(LPCSTR lpcszStr, LPWSTR lpwszStr, DWORD dwSize)
{
    // Get the required size of the buffer that receives the Unicode
    // string.
    DWORD dwMinSize;
    dwMinSize = MultiByteToWideChar(CP_ACP, 0, lpcszStr, -1, NULL,
0);
    assert(dwSize >= dwMinSize);

    // Convert headers from ASCII to Unicode.
    MultiByteToWideChar(CP_ACP, 0, lpcszStr, -1, lpwszStr,
dwMinSize);
    return TRUE;
}

void play(string s)
{
    wchar_t wText[30] = { 0 };
    MByteToWChar(s.c_str(), wText, sizeof(wText) / sizeof(wText[0]));
    PlaySound(wText, NULL, SND_FILENAME | SND_SYNC);
}
```

```
AddressBook::AddressBook()
{
    string str;
    count = 0;
    ifstream read("data.txt");
    read >> str;
    if (str.length() == 0)
    {
        read.close();
        return;
    }
    int p1 = 0, p2 = 0, i = 0;
    while (p2 != (str.length() - 1))
    {
        p1 = str.find('{', p2);
        p2 = str.find('}', p1 + 1);
        Contact[i++].Read(str, p1, p2);
        Contact[i - 1].setNumber(i);
    }
    count = i;
    read.close();
}

AddressBook::~AddressBook()
{
    ofstream Save("data.txt");
    for (int i = 0; i<count; i++)
    {
        Save << '{';
        Contact[i].Save(Save);
        Save << '}';
    }
    Save.close();
}

void AddressBook::Add()
{
    int k;
    k = count++;
    Contact[k].add(k);
}

void AddressBook::Edit(int i)
```

```
{
    Contact[i - 1].Edit();
}

void AddressBook::Delete(int i)
{
    Contact[--i].Delete(); count--;
    for (i; i < count; i++)
    {
        contact temp = Contact[i];
        Contact[i] = Contact[i + 1];
        Contact[i + 1] = temp;
        Contact[i].setNumber(i + 1);
    }
}

bool AddressBook::print(int i)
{
    if (i<1 || i>count)
    {
        cout << "                查无此人！\n";
        cout << "                ";
        system("pause");
        system("cls");
        return false;
    }
    cout << "
**********************************************************************
***************\n";
    Contact[i - 1].print();
    cout << "
**********************************************************************
***************\n\n\n";
    return true;
}

bool AddressBook::printAll()
{
    if (count == 0)
    {
        cout << "                联系人为空，请新建联系人！\n";
        cout << "                ";
        system("pause");
        system("cls");
```

```
        return false;
    }
    cout << "
**********************************************************************
***************\n";
    for (int i = 0; i < count; i++)
        Contact[i].print(1);
    cout << "
**********************************************************************
***************\n";

    return true;
}

void contact::print()
{
    cout << "                编号：" << Number << "  姓名：" << Name
<< "  单位：" << Company
        << "  电话：" << PhoneNumber << "  邮箱：" << Email << "  分
类：" << Classification << endl;
}

void contact::print(int i)
{
    cout << "                编号：" << Number << "  " << "姓名：" <<
Name<< endl;
}

void contact::Save(ofstream& save)
{
    save << '*' << Name << '*' << Company << '*' << PhoneNumber
        << '*' << Email << '*' << Classification << '*';
}

void contact::Edit()
{
    cout << "                输入修改后的信息！\n";
    cout << "                编号为：" << Number << endl;
    cout << "                姓名：";
    cin >> Name;
    cout << "                单位：";
    cin >> Company;
    cout << "                电话:";
    cin >> PhoneNumber;
```

```
    cout << "                邮箱:";
    cin >> Email;
    cout << "                分类:";
    cin >> Classification;
}

void contact::add(int n)
{
    cout << "                输入新增联系人的信息：\n";
    Number = n + 1;
    cout << "                编号为：" << Number << endl;
    cout << "                姓名：";
    cin >> Name;
    cout << "                单位：";
    cin >> Company;
    cout << "                电话:";
    cin >> PhoneNumber;
    cout << "                邮箱:";
    cin >> Email;
    cout << "                分类:";
    cin >> Classification;
}

void contact::Delete()
{
    Number = 0;
    Name = "null";
    Company = "null";
    PhoneNumber = "null";
    Email = "null";
    Classification = "null";
    cout << "                删除成功！\n";
}

void contact::Read(string Str, int k1, int k2)
{
    int r1 = k1 + 1, r2 = k1 + 1, i = 0;
    while (r2 != (k2 - 1))
    {
        r1 = Str.find('*', r2);
        r2 = Str.find('*', r1 + 1);
        switch (i++)
        {
        case 0:Name.assign(Str, r1 + 1, r2 - r1 - 1); break;
```

```
        case 1:Company.assign(Str, r1 + 1, r2 - r1 - 1); break;
        case 2:PhoneNumber.assign(Str, r1 + 1, r2 - r1 - 1); break;
        case 3:Email.assign(Str, r1 + 1, r2 - r1 - 1); break;
        case 4:Classification.assign(Str, r1 + 1, r2 - r1 - 1); break;
        }
    }
}

contact& contact::operator=(contact& C)
{
    this->Name = C.Name;
    this->Company = C.Company;
    this->PhoneNumber = C.PhoneNumber;
    this->Email = C.Email;
    this->Classification = C.Classification;
    return *this;
}
```