课程设计报告

项目题目： 通讯录管理系统

学生姓名： 陈建宇

学 号： 20202132045

专 业： 网络工程

班 级： 2020级2班

**(1)题目**

通讯录管理系统

**目的：**设计一个实用的小型通讯录管理系统，实现用户方便查找自己所需的信息。

**系统主要功能：**

1. 录入：记录信息包括姓名、电话（可自定义其他信息）；
2. 更新：增加、删除记录；
3. 显示所有记录；
4. 查询记录：根据姓名、电话查询；
5. 退出。

设计要求：

1. **数据存储在文件中。**

**2. DOS版/WINDOWS版(WINDOWS版难度系数高)。**

**3．使用类、向量（参看文献VC的STL）。**

**4. WINDOWS版的使用的是VC MFC模式，关键技术：消息机制、窗口类、对话框与控件、菜单。**

**(2)前言**

现在看来独立的通讯录似乎不再像多年前那么使用了，我们已经习惯使用通讯软件内置的通讯录(如QQ、微信等)，但在信息时代到来之初，通讯录确实我们社交、学习、工作不可或缺的工具，毕竟在相关技术和服务不那么完善和成熟的时候，有些信息还是需要自己动手记录。

此通讯录系统采用Visual Studi0 2019编程，实现一个命令行运行窗口界面，并能实现联系人的添加、编辑以及删除，对联系人的信息进行查找，写入文件。

1. **系统设计思路**

### 2.1系统执行流程

先是对程序的预处理，然后初始化各个类的信。然后，根据用户的需要，对通讯录里的联系人信息进行编辑。

对联系人信息可以进行添加、编辑、删除、查询操作，也可将信息保存在

文件中。

### 2.2 源文件构成

头文件:

Contact.h：存放Contact类的声明

Address\_List.h：存放Address\_List类的声明

Menu.h：存放菜单函数声明

源文件:

Contact\_Function.cpp：存放Contact类函数的实现

Address\_Function.cpp：存放Address\_List类函数的实现

Main\_Menu.cpp：存放菜单相关函数的实现

### 2.3 容错处理

对数据的有效性进行判断。然后，对一些数据输入合法性进行检验。

对敏感操作的二次确认。在进行删除、清空和退出等有可能丢失数据的操作时，为避免用户的误操作，需要进行二次确认。

1. **系统的设计过程：具体描述**

**每一个类**

1. Contact类:

用于存放联系人数据

私有成员变量:姓名，年龄，性别，电话，邮箱，备注

静态成员变量:item\_type,用于存放数据种类，

公共成员函数:

void Input():从输入流中提取字符串为成员变量赋值

void Print():将当前Contact对象的成员变量输出至屏幕

string getName():返回成员变量Name的值

string getPhone\_Number():返回成员变量Phone\_Number的值

友元函数

friend ostream& operator<<(ostream& os, Contact& contact\_Argu):用于向文件输出数据

1. Address\_List类:

成员变量:vector<Contact>,用于顺序存放联系人

成员函数:

bool isEmpty()const:判断当前联系人列表是否为空

void AppendRecord(int opt\_size = 1):新录入联系人

void EmplaceRecord(int index, Contact contact\_argv); // (插入位置，传入数据)

void DeleteRecord(int index = -1, int opt\_size = 1); //删除 -1-尾部 1-1个

void Display(int index = -1, int opt\_size = 1)const; //显示 -1-全部 1-1个

int Inquiry(const string& pattern, int search\_mode = 1)const;:查询联系人

void ClearRecord();//清空

//文件操作

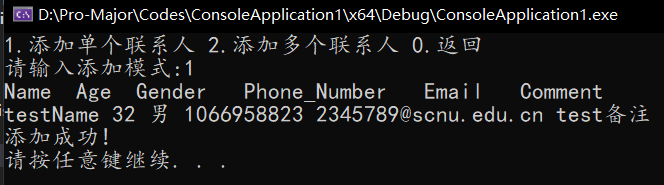
void Save\_As\_Txt(const string fileName = "AddressList\_Save1.txt");//写入文件

void Read\_As\_Txt(const string fileName = "AddressList\_Save1.txt");//读取文件

1. **系统测试情况**
2. **主菜单样式**



1. **添加联系人**



1. **删除联系人**

void Address\_List::DeleteRecord(int index, int opt\_size)

{

if (index == -1)

Persons.pop\_back();

else // if(index >= 0)

{

vector<Contact>::iterator ptr = Persons.begin()+ index;

Persons.erase(ptr, ptr + opt\_size);

}

}

1. **查找联系人**

通过联系人姓名查找

函数返回目标联系人下标

int Address\_List::Inquiry(const string& pattern, int search\_mode)const

{

if (search\_mode == 1)

{

for (int i = 0; i < Persons.size(); ++i)

{

if (pattern == Persons[i].getName())

return i;

}

return -1;

}

else//if(search\_mode == 2)

{

for (int i = 0; i < Persons.size(); ++i)

{

if (pattern == Persons[i].getPhone\_Number())

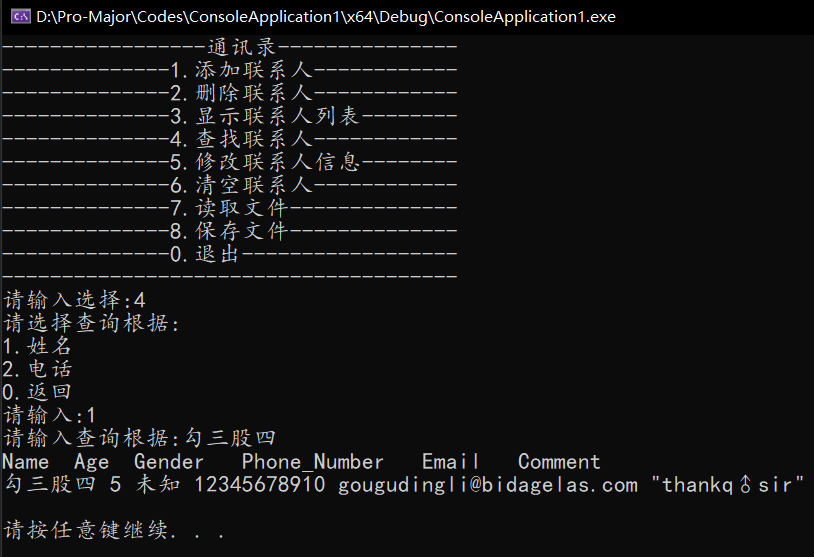
return i;

}

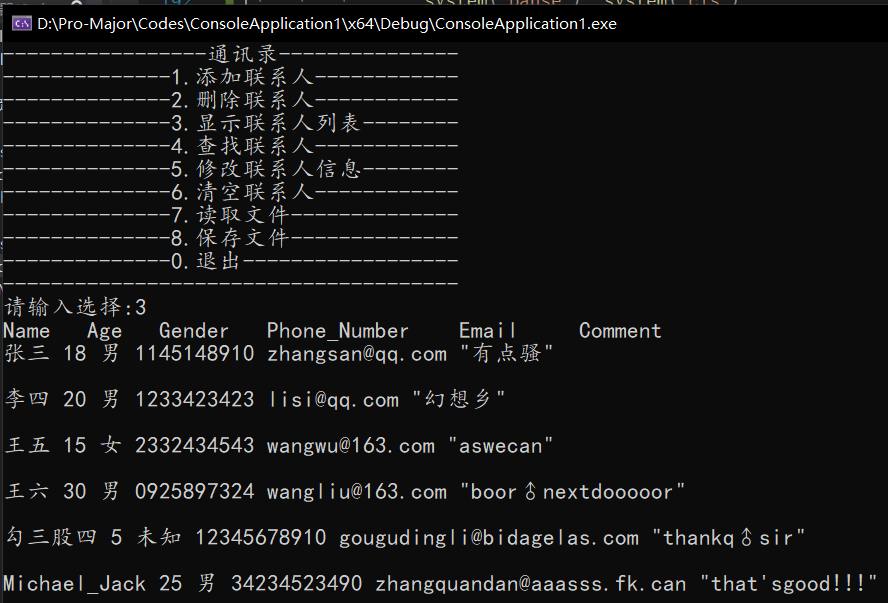
return -1;

}

}



1. **显示联系人**



1. **修改联系人信息**

void Modificate\_Mode(Address\_List& list1)

{

cout << "请输入要修改的联系人姓名:";

string pattern;

cin >> pattern;

int pos = list1.Inquiry(pattern, 1);

if (pos == -1)

{

cerr << "查无此人!" << endl;

return;

}

char choice5;

while (true)

{

cout << "1.Name" << endl

<< "2.Age" << endl

<< "3.Gender" << endl

<< "4.Phone\_Number" << endl

<< "5.Email" << endl

<< "6.Comment" << endl

<< "0.返回" << endl

<< "请输入要修改的项:" << endl;

cin.ignore(std::numeric\_limits< streamsize >::max(), '\n');

choice5 = cin.get();

cin.ignore(std::numeric\_limits< streamsize >::max(), '\n');

string substi;

switch (choice5)

{

case '0': cout << "即将返回..." << endl; return; break;

case '1': cout << "Name:"; cin >> substi; list1.ModRecord(substi, pos, 1); cout << "修改成功!" << endl; break;

case '2': cout << "Age:"; cin >> substi; list1.ModRecord(substi, pos, 2); cout << "修改成功!" << endl; break;

case '3': cout << "Gender:"; cin >> substi; list1.ModRecord(substi, pos, 3); cout << "修改成功!" << endl; break;

case '4': cout << "Phone\_Number:"; cin >> substi; list1.ModRecord(substi, pos, 4); cout << "修改成功!" << endl; break;

case '5': cout << "Email:"; cin >> substi; list1.ModRecord(substi, pos, 5); cout << "修改成功!" << endl; break;

case '6': cout << "Comment:"; cin >> substi; list1.ModRecord(substi, pos, 6); cout << "修改成功!" << endl; break;

default:cerr << "输入有误,请重新输入!" << endl; system("pause"); system("cls"); break;

}

}

}

1. **清空联系人列表**

void Address\_List::ClearRecord()

{

Persons.clear();

if (!Persons.empty())

{

cerr << "Eeception:Address\_List not empty yet!" << endl;

abort();

}

}

1. **保存文件**

以文本文件形式保存(已有以二进制形式储存，但未调用)

无论指定储存文件是否存在，若要修改，可读入内存后进行修改，再存入文件，写文件模式为舍弃原有内容

文件默认保存路径与可执行文件所在目录相同

void Address\_List::Save\_As\_Txt(const string fileName)

{

ofstream ostrm;

ostrm.open(fileName, ios::out | ios::trunc);

if (!ostrm.is\_open())

{

cerr << fileName << "Error:Open Fail!" << endl;

ostrm.close();

return;

}

for (int i = 0; i < Persons.size(); ++i)

ostrm << Persons[i] << endl;

ostrm.close();

}

1. **读取文件**

若储存文件存在，则程序开始时先从文件读入数据。

void Address\_List::Read\_As\_Txt(const string fileName)

{

ifstream istrm;

istrm.open(fileName, ios::in | ios::beg);

if (!istrm.is\_open())

{

cerr << fileName << "Error:Open Fail!" << endl;

istrm.close();

return;

}

while (!istrm.eof())

{

Persons.emplace\_back();

string tmp;

for (int i = 1; i < Contact::item\_type + 1; ++i)

{

istrm >> tmp;

Persons.back().Modificate(tmp, i);

}

}

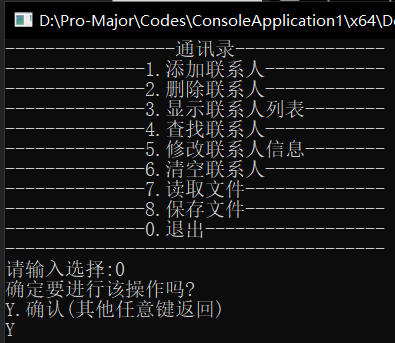
istrm.close();

return;

}

1. **退出**

需输入大写字母Y才能退出，任意其他输入无效



**(6) 系统的优点与改进**

优点:

1. 一定程度上优化了人机交互界面，增加了用户操作的容错率
2. 采用了C++中STL容器库的vector数组存放数据，对于添加、删除等操作较为方便

改进:

1. 通讯录中数据种类较少,且用户无法自定义
2. 用户在录入联系人信息时，无法缺省某些项
3. 由于采用了控制台窗口，交互界面相比对话框仍逊色不少
4. 除了基本功能外，额外辅助功能，如排序、分组等较少
5. 用户无法自定义文件名和存储路径
6. 程序容错率仍然较低，且存在未知的但不影响主要功能的错误（如输入选项时会出现一次”吞字符“的情况，即输入明明符合要求，却被无视并要求重新输入）

**(7) 感想**

编写程序并不是一件容易的事，尤其是个人要开发出一个相对完善的程序。编写代码，不仅要熟练掌握所使用的语言的语法规则，还需要宏观布局能力，能设计好项目的框架和结构，构思各种功能的作用，想办法实现这些功能，以及用什么算法实现。

程序错误一直令人十分头疼。语法错误尚能接受，编译器一般能够比较准确地检查出来，但是逻辑错误则很难被发现。通常，一个中型程序代码量是相当可观的，要从这么多代码中找到一个小错误是比较困难的，这需要很大的耐心。本人在开发通讯录管理系统时，遇到了不少难以察觉的错误，不过好在不少IDE都包含较为完善的调试功能。所以，身为程序员，有耐心和有逻辑推理能力是必不可少的素质。

开发一个小程序，程序员往往得自己编写程序的界面。而界面如何布局才能做到美观紧凑，合理大方，这又是程序员需要面临的一大难题。程序的界面应布局整齐，这是最基本的要求。一来美观，二来方便用户使用。布局的另一个要求便是紧凑，空余的空间尽量减少，发挥程序的最大功能。这就要求程序员有足够的审美能力和布局能力。

有关技术文档阅读:

1. 优先阅读官方文档

这可以说是第一手资料，而非他人消化后给出的第二首资料，如果别人的理解有失偏颇，正好将仍是“小白”的你导入了错误的方向，那这将浪费不少时间，降低学习的效率。

1. 首先把握文档整体架构

一般来说，官方文档，体量可能稍大，甚至庞杂，因此有必要提前有个整体印象。

通常，文档首部分都是由介绍或引言组成，阅读它可以帮助读者快速对文档产生大概的印象，了解该技术或产品的基本功能和作用。再者，若是比较明确目标段落，通过目录也可快速定位。

1. 不惧怕英文文档