

**院 系：计 算 机 学 院**

**实验课程：程序设计项目课程**

**实验项目：通讯录管理系统**

**指导老师：宋桂琴**

**开课时间：2021 ～ 2022年度第 1学期**

**专 业：计算机科学与技术**

**班 级：一班**

**学 生：莫崇浩**

**学 号：20202131062**

**华南师范大学教务处**

1. **题目**

**通讯录管理系统（使用了Qt实现）**

**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。**

1. **前言**

这次项目使用了Qt这一以C++为基础的应用程序开发框架，其实之所以所以使用Qt是想学点新东西， 身为计算机学生，我们应该熟悉关于窗口化编程的使用。

本次课程设计的目的是让学生以各自的方式来编写出一个系统，并对windows窗口程序编程进行一定的学习和研究，如窗口类，对话框，按钮等控件的使用。

1. **系统设计思路**

1、分析需求：首先分析一下项目的需求，有数据的展示，添加，修改和删除功能，还有查询功能，我在展示中本来是想要做成手机一样以名字排序（汉字首字的拼音首字母），但是最后放弃了，就做了一个简单一点的单纯的字符串比较和手机号码的大小比较。

2、根据需求创建窗口函数：有多少需求就需要多少窗口， 但是要是多个窗口管理起来有些凌乱，所以我就想在选择功能的时候关闭选择窗口，打开功能窗口，然后在功能窗口塞个返回键返回主菜单。

3、完善窗口功能：在其中编写功能窗口中所需功能的函数，并与窗口中控件相连接，以形成完整的窗口

4、测试

1. **系统设计过程**
2. 首先创建一个Qt的空项目并命名为AddressList，因为我的程序的基本是一窗口类widget有关，所以在Qt项目的pro文件中需要加入

QT += widgets

表示引入了QtWidget这个模块（module）

1. 创建所需窗口，有菜单窗口menu、添加数据窗口addperson、修改数据窗口changed、查询窗口selectperson、展示窗口showandsort。创建窗口并非需要详述的事，就不一一展示了
2. 将各个窗口所需控件用其ui的设计界面设计好，如下图

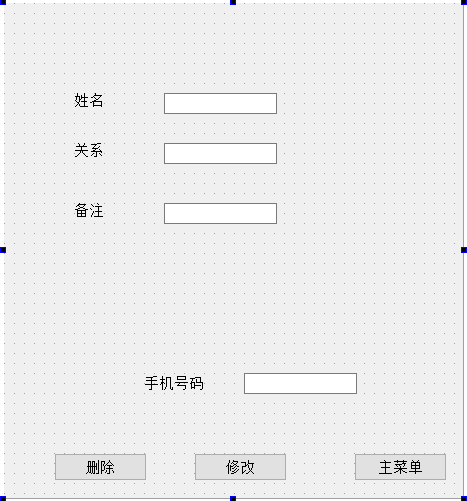
菜单



添加数据窗口



修改数据窗口



查询数据窗口



查看（排序）窗口



1. 为了实现窗口间的切换，我查询了一些资料，我觉得有两种方式可以实现，一是通过在主函数中将其全部窗口展现，但是使用hide函数来将菜单以外的窗口隐藏，然后通过按钮的信号与其他窗口的show函数和hide函数形成的槽组成一组信号槽来组成功能，二是从网上查来比较巧妙的方法，需要运用到一个新的函数QStackedLayout，创建一个主窗口mainWidget并且把子窗口，这个可以使多个widget在同一个布局中，要切换显示的部件使用setCurrentIndex(传入参数为下标)，或者使用setCurrentWidget(传入参数为部件的指针)。设置当前部件就是将该部件置于栈顶。然后用setLayout将主窗口设为唯一布局，这样的优点是所有窗口会在同一个窗口显示，所以能显示出的就只有一个，满足所需功能。我选择了二。

#include "mainwidget.h"

#include "ui\_mainwidget.h" //用来实现窗口的转换

mainwidget::**mainwidget**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::mainwidget)

{

ui->setupUi(this); //在此窗口创造各种子窗口的空间

this->setMinimumSize(700,600); //固定窗口大小

this->setMaximumSize(700,600);

this->setWindowTitle("通讯录管理系统"); //由于所有子窗口等于都是在主窗口中打开，所以它的大小和标题名字都是Mainwidget的

menuwidget=new Menu;

addpersonwidget=new AddPerson;

changewidget=new Changed;

selectpersonwidget=new selectperson;

showandsortwidget=new showandsort;

stackedLayout=new QStackedLayout;

stackedLayout->addWidget(*menuwidget*); //将创造的widget窗口添加到布局当中（基本用法）

stackedLayout->addWidget(*addpersonwidget*);

stackedLayout->addWidget(*changewidget*);

stackedLayout->addWidget(*selectpersonwidget*);

stackedLayout->addWidget(*showandsortwidget*);

setLayout(*stackedLayout*);//可以重新父化此布局中的部件（也就是设置唯一布局）（实际上多个界面堆叠到了一个窗口内）

connect(menuwidget, &Menu::display, stackedLayout, &QStackedLayout::setCurrentIndex);

//setCurrentIndex此属性保存小部件可见的索引位置

//如果当前没有小部件，则当前索引为-1

connect(addpersonwidget,&AddPerson::display,stackedLayout,&QStackedLayout::setCurrentIndex);

connect(changewidget,&Changed::display,stackedLayout,&QStackedLayout::setCurrentIndex);

connect(selectpersonwidget,&selectperson::display,stackedLayout,&QStackedLayout::setCurrentIndex);

connect(showandsortwidget,&showandsort::display,stackedLayout,&QStackedLayout::setCurrentIndex);

}

connect函数是Qt基本中的基本，是信号与槽的基础，这里connect（窗口对象，关于窗口的信号，布局，布局中的槽函数）

setCurrentIndex此属性保存小部件可见的索引位置

这些connect的都需各自在各个窗口中创一个signal：display函数来当做一个信号

1. 完善通讯录中人类的类，将我们所要数据放进类中，分别为姓名name，手机号码number，关系relation，备注nickname，还有其各自的get和set函数

**PersonInfo**(QString newname,int newnumber,QString newrelation,QString newnickname)

{

name=newname;

number=newnumber;

relation=newrelation;

nickname=newnickname;

}

QString **getName**()

{ return name;}

QString **getRelation**()

{return relation; }

QString **getNickname**()

{return nickname; }

int **getNumber**()

{return number;}

void **setName**(QString newname)

{name=newname; }

void **setRelation**(QString newrelation)

{relation=newrelation;}

void **setNickname**(QString newnickname)

{nickname=newnickname; }

void **setNumber**(int newnumber)

{number=newnumber; }

6、完善添加函数，给按钮的点击信号完善其函数，一个为返回键，一个为添加键，返回键的函数只是一个信号emit display（0），添加键的较为重要，这些函数代表此键被某种信号触发后会引发的处理。

void AddPerson::**on\_addpushButton\_clicked**()//添加 按钮点击

{

QString name=ui->namelineEdit->text();

//将用户输入的名字从nameLineEdit取出

int number =ui->numberlineEdit->text().toInt();

//将用户输入的电话号码从numberlineEdit中取出并转为整形

QString relation=ui->relalineEdit->text();

//将用户输入的关系从relalineEdit取出

QString mark=ui->marklineEdit->text();

//将用户输入的备注从marklineEdit中取出

if(ui->namelineEdit->text()==""||ui->numberlineEdit->text()==""||ui->relalineEdit->text()==""||ui->marklineEdit->text()=="")

{

QMessageBox::about(NULL,"错误","存在空项！");

//原型：static void about(QWidget \*parent, const QString &title, const QString &text);

//QMessageBox::about(NULL,"About","About this application");这三个参数分别队友父窗口对象，对话框标题和对话框内容

return;

}

QFile file("person.txt");//实例化一个文件person.txt来保存信息

file.*open*(QIODevice::WriteOnly|QIODevice::Text|QIODevice::Append);

//QIODevice类是Qt中所有I/O设备的基本接口类。

//QIODevice为了支持读写块数据，提供了一个共同的实现和一个通用的接口，这些块数据如： QFile, QBuffer 和 QTcpSocket.

//open()可以用来打开文件这里QIODevice::WriteOnly代表将文件以只写的方式打开

//QIODevice::Text代表我们打开的是文本文件，QIODevice::Text会对end-of-line结束符进行转译

//QIODevice::Append以追加的方式打开，新增加的内容将被追加到文件末尾

if(!file.isOpen())//如果数据文件没有打开，弹开对话框提示

{

QMessageBox::about(NULL,"错误","文件打开失败");

return;

}

QTextStream out(*&file*);

//QTextStream可以进行一些基本的文本读写

out<<name<<" "<<number<<" "<<relation<<" "<<mark;

out<<Qt::endl;

//写入文件

file.*close*();

QMessageBox::about(NULL,"提示","插入成功");

ui->namelineEdit->clear();

ui->numberlineEdit->clear();

ui->relalineEdit->clear();

ui->marklineEdit->clear();

//将用户输入的数据清空

}

7、完善修改信息，和6一样，重要在于删除键的函数和修改键的函数，由于两者原理一样，以下就只放删除键的函数，但因为这两者需要改变数据组本身而且最好不要改变其原来顺序，这需要在读取中用一个容器去保存并修改其数据，且并不影响顺序。这里使用了QVector也就是向量来读取文件

void Changed::**on\_delepushButton\_clicked**()//删除数据的按钮点击函数

{

//因为需要搜查文件内数据，所以需要一个容器保存

QVector<PersonInfo> allPersonInfo; //数据类型为PersonInfo的QVector容器

if(ui->numberlineEdit->text()=="")

{

QMessageBox::about(NULL,"错误","手机号码不得为空！");

return;

}

QFile file("person.txt");

file.*open*(QIODevice::ReadOnly|QIODevice::Text);

if(!file.isOpen())

{

QMessageBox::about(NULL,"错误","文件无法打开！");

return;

}

QTextStream input(*&file*);

//以file作为Qt文件流

while(!input.atEnd())

{

QString name,relation,mark;

int number;

input>>name>>number>>relation>>mark;

//读入名字，手机号码，关系，备注

allPersonInfo.push\_back(PersonInfo(name,number,relation,mark));

//调用之前建立的构造函数从而函数实例化一个对象PersonInfo并将其加入allPersonInfo

}

allPersonInfo.pop\_back(); //文件最后会多读一个无用数据，将其拿出？？

file.*close*();

int number=ui->numberlineEdit->text().toInt();

//获取用户输入的手机号码

bool flag=false;

for(QVector<PersonInfo>::iterator it=allPersonInfo.begin();it !=allPersonInfo.end();it++)

{

if(it->getNumber()==number)

{

allPersonInfo.erase(it);

flag=true;

break;

}

}

if(flag==true)

{

QMessageBox::about(NULL,"提示","删除成功");

file.*open*(QIODevice::WriteOnly|QIODevice::Text|QIODevice::Truncate);

//QIODevice::Truncate在写入时会从文件开始处写入，覆盖原有内容

//以只写覆盖的方式打开文本文件

if(!file.isOpen())

{

QMessageBox::about(NULL,"错误","数据文件无法打开");

return;

}

QTextStream out(*&file*);

for(auto i: allPersonInfo)//c++11的新特性,i用来遍历allPersonInfo中所有元素

{

out<<i.getName()<<" "<<i.getNumber()<<" "<<i.getRelation()<<" "<<i.getNickname();

out<<Qt::endl;//Qt 6.0版本的改动，无法直接使用endl(这里查了好久。。。)

}

file.*close*();

}

else

{

//如果未进行修改，弹出不存在对话框

QMessageBox::about(NULL,"错误","手机号码不存在！");

}

//关闭文件

ui->numberlineEdit->clear();

//清空所有输入框

}

8、完善查询信息，原理与7中的删除功能函数一样，甚至简单一点，也是要存一遍文本数据，存到向量，然后根据输入的手机号码去一个个查找并显示所查的数据

void selectperson::**on\_selectpushButton\_clicked**()

{

if(ui->numberlineEdit->text()=="")

{

QMessageBox::about(NULL,"错误","手机号码不能为空");

return;

}

QFile file("person.txt");

file.*open*(QIODevice::ReadOnly|QIODevice::Text);

if(!file.isOpen())

{

QMessageBox::about(NULL,"错误","数据文件打开失败");

return;

}

QTextStream input(*&file*);

//以file作为Qt文本流

QVector<PersonInfo>allPersonInfo;

//数据类型为PersonInfo的QVector容器

while(!input.atEnd())

{

QString name,relation,mark;

int number;

input>>name>>number>>relation>>mark;

allPersonInfo.push\_back(PersonInfo(name,number,relation,mark));

//调用之前建立的构造函数实例化一个PersonInfo对象并将其加入allPersonInfo

}

allPersonInfo.pop\_back();

//丢掉最后的无用数据

file.*close*();

//关闭文件

int number=ui->numberlineEdit->text().toInt();

bool flag = false; //为检查立一个bool标记记录检查状况

for(auto i:allPersonInfo)

{

if(i.getNumber()==number)

{

ui->shownamelabel->setText(i.getName());

ui->showrelalabel->setText(i.getRelation());

ui->showmarklabel->setText(i.getNickname());

flag=true;

break;

}

}

if(!flag)

{

QMessageBox::about(NULL,"错误","number不存在");

}

ui->numberlineEdit->clear();

}

9、完善展示和搜索功能，展示中就需要一个表格，所以在ui中将tableWidget初始化，在点击按钮中同样直接读取文件，但是用了一个取巧的方法，我使用了c++算法头文件中的排序。

showandsort::**showandsort**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::showandsort)

{

ui->setupUi(this);

sortButtonGroup=new QButtonGroup;

//实例化sortButtonGroup

sortButtonGroup->addButton(*ui->nameradioButton*,0);

//将代表名字的单选框加入sortButtonGroup并将其id设为0

//不建议使用默认id，默认id是由-2开始递减

sortButtonGroup->addButton(*ui->numberradioButton*,1);

//将代表数字的单选框加入sortButtonGroup并将其id设为1

ui->nameradioButton->setChecked(true);

//名字的选项默认选中

ui->tableWidget->setColumnCount(4);

QStringList headerLabel;

//用来记录表格表头数据

headerLabel<<tr("姓名")<<tr("手机号码")<<tr("关系")<<tr("备注");

ui->tableWidget->setHorizontalHeaderLabels(headerLabel);

// void setHorizontalHeaderLabels(const QStringList &labels);设置水平表头标签

ui->tableWidget->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

//设置根据总宽度调整列宽

ui->tableWidget->setRowCount(1);

//初始行数设为一行

}

bool **cmp\_name**(PersonInfo a,PersonInfo b)

{

return a.getName()<b.getName();

}

bool **cmp\_number**(PersonInfo a,PersonInfo b)

{

return a.getNumber()<b.getNumber();

}

void showandsort::**on\_sortpushButton\_clicked**()

{

ui->tableWidget->clearContents();

//清空除表头外的所有数据

//若想清除表头可使用clear（）

int sortTag=sortButtonGroup->checkedId();

//取得当前选中的单选框的id

QVector<PersonInfo>allPersonInfo;

//数据类型为PersonInfo的QVector容器

if(!getPersonInfo(*allPersonInfo*))//从文件读取到allPersonInfo

return;

if(allPersonInfo.empty())

{

QMessageBox::about(NULL,"错误","没有学生信息");

return;

}

switch(sortTag)

{

case 0: //sortTag==0根据名字进行排序

std::sort(allPersonInfo.begin(),allPersonInfo.end(),cmp\_name);

break;

case 1: //sortTag==1根据手机号码进行排序

std::sort(allPersonInfo.begin(),allPersonInfo.end(),cmp\_number);

break;

}

ui->tableWidget->setRowCount(allPersonInfo.size());

//将表格的行数设为与学生数量相同

//否则setItem添加单元格不会生效

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

ui->tableWidget->setItem(i, 0, new QTableWidgetItem*(allPersonInfo[i].getName())*);

ui->tableWidget->setItem(i, 1, new QTableWidgetItem*(*QString*::number(allPersonInfo[i].getNumber()))*);

ui->tableWidget->setItem(i, 2, new QTableWidgetItem*(allPersonInfo[i].getRelation())*);

ui->tableWidget->setItem(i, 3, new QTableWidgetItem*(allPersonInfo[i].getNickname())*);

//设置各单元格的数据，非QString类型的数据需要转换为QString

}

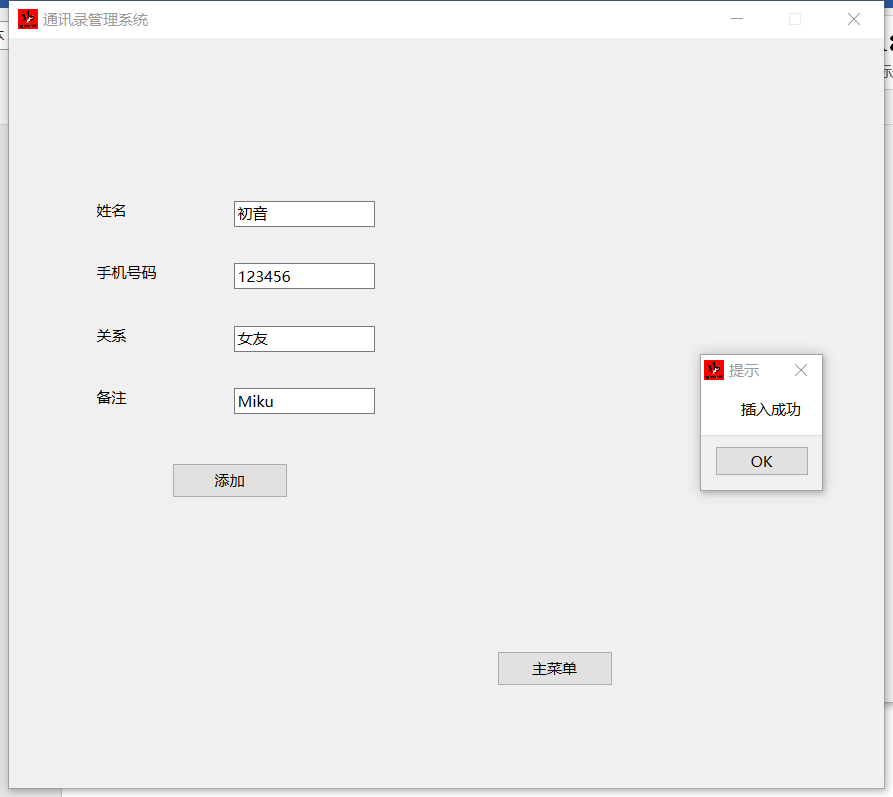
}

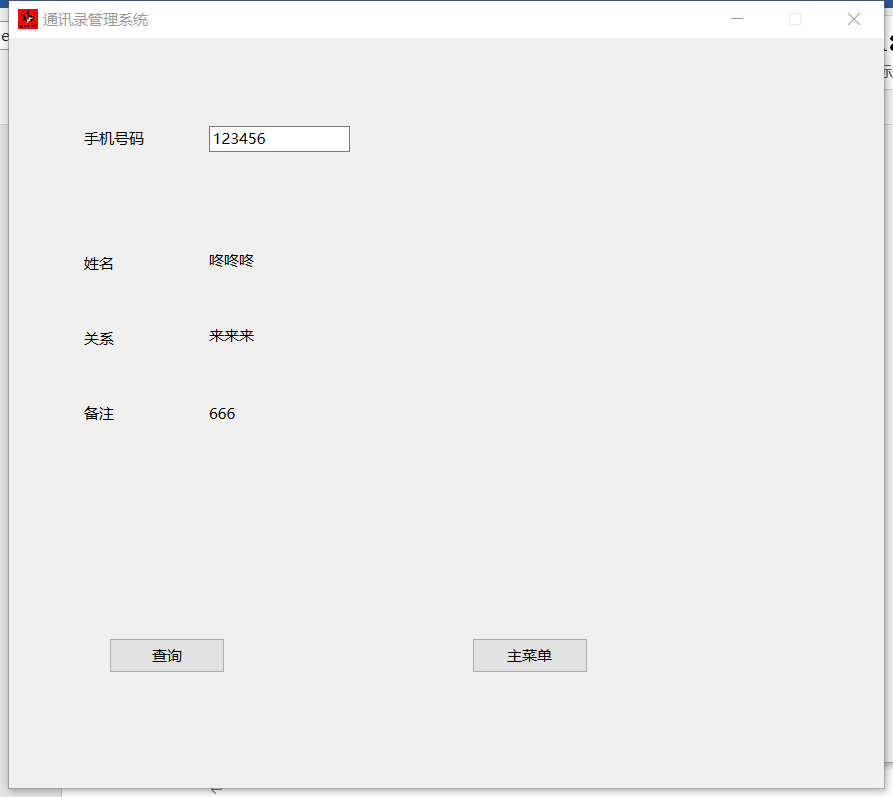
1. **系统测试情况**

运行情况，如下图所示

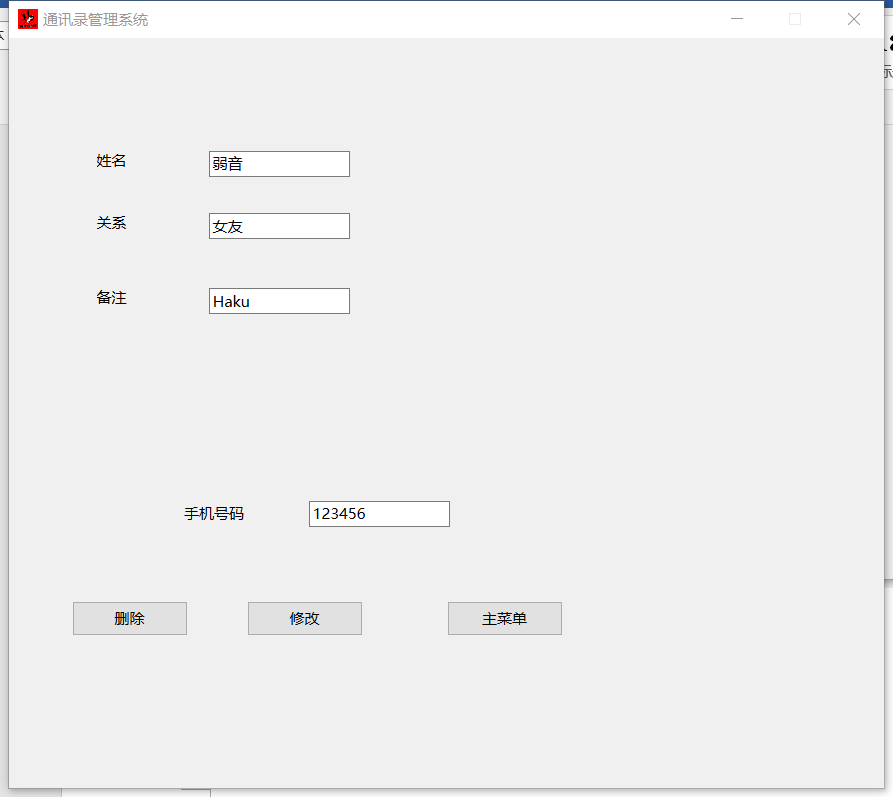


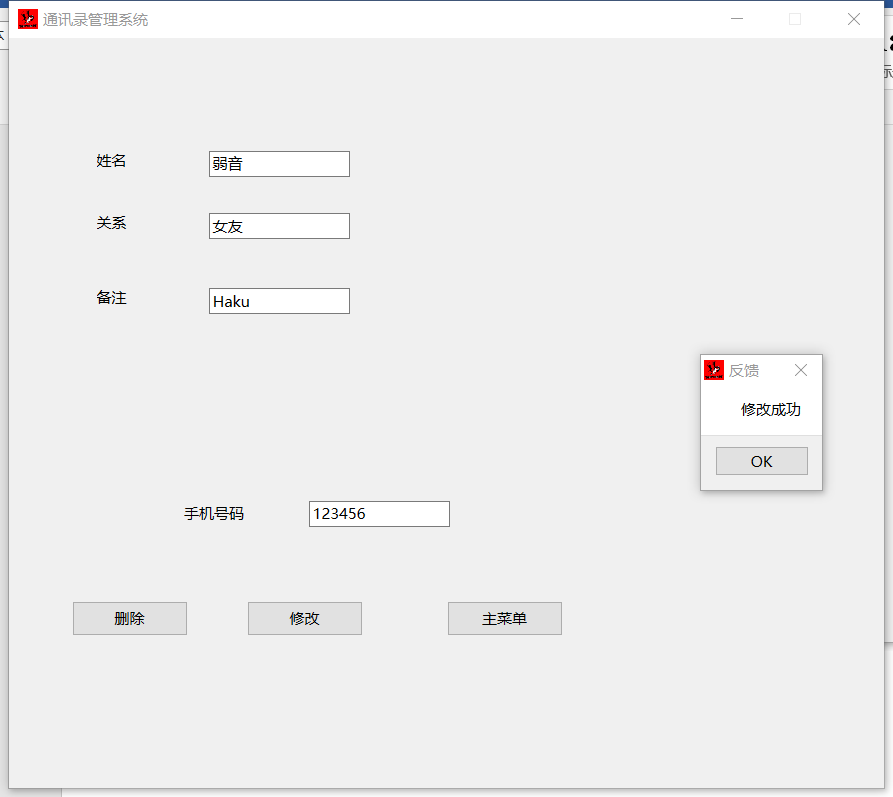




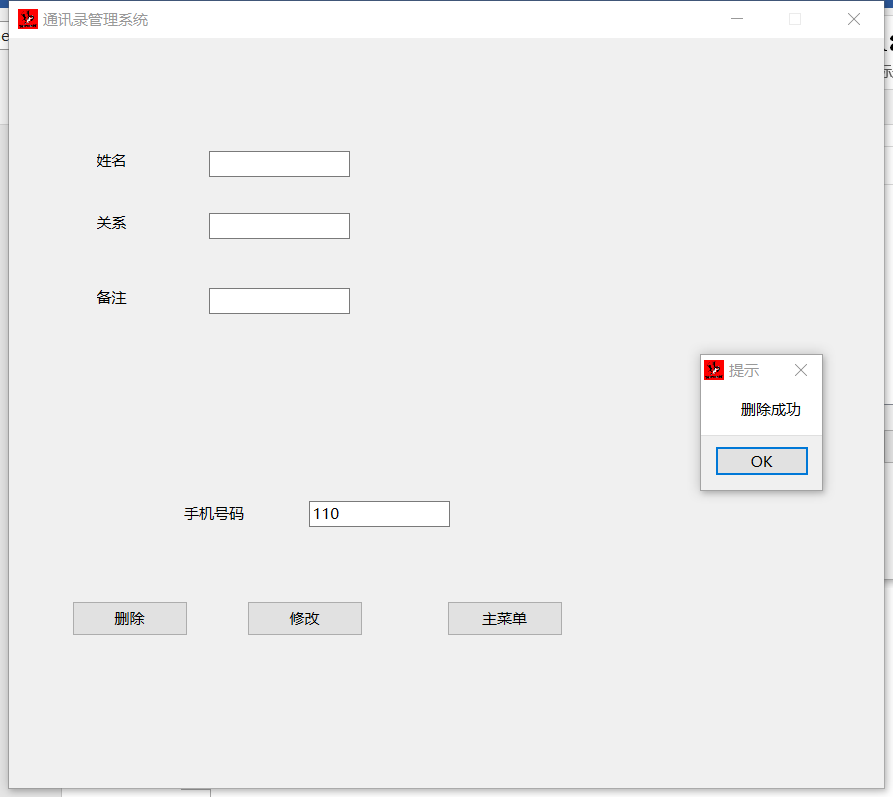














1. **系统的优点与改进**

优点：实现了窗口一体化，在有错误输入时，会给予提示并阻止函数进行，一次只显示一窗口使系统感觉上更加一体化

需改进的：一、在切换窗口是没有使之前的窗口刷新，文本框中会保留原本输入但没执行的数据和查询与查看过的显示数据

二、没有写删除所有或者刷新文件的函数，在删除时会比较费力

1. **感想**

在使用Qt时，我虽然买了本书且看了一点视频，但是本身项目却很难一下子有想法，而且Qt的错误比较难找，搜索渠道上关于Qt的资料比较少，

在自己摸索的时候，经常会出现找不到的错误，而且Qt的报错方式比较难确定错误，导致我其实是根据一个成型的系统去进行模仿与参考的。但是即使是有参考也一定要去理解他所利用的所有东西，看到没用过的函数就要去查并且理解为什么要用他。Qt本身是一个比较方便的窗口化编程框架，但是学习成本其实是比较高的，因为他的库中很丰富，但又比较杂碎，而且他的函数指导书虽然可以随时打开，但是全是英文，而我是个英语比较差的。还有一种比较难受的情况就是，因为我下载的是Qt的最新版本也就是1.6版本，和1.5版本有些出入，但是在编写中，你出错了，这种错又查不出来，因为新版本的回答和问题很少，就很难受，比如在输出流需要换行时，原本可以使用endl的，但是1.6中是要Qt::endl，这种错误难以避免，却难以解决。

通过这次项目的编写，使我学到了不少实用的知识，跟重要的是，这个过程，会让我对窗口化编程的流程更加理解并熟悉，在做其他类似东西的时候也可以更加得心应手。