

C语言程序设计课程设计报告

系统名称 个人信息管理系统

|  |  |
| --- | --- |
| **学院：** | **计算机与电子信息学院** |
| **班级：** | **计算机类222班** |
| **学号：** | **2207310209 2207310203** |
| **姓名：** | **刘军男 黄虹霖** |
| **指导老师：** | **杨柳** |

1. 系统简介

在日常生活中，各类app、网站为我们带来了更为便捷的生活方式，其中，个人信息管理系统是必不可缺的一部分。为了解决传统管理方式繁琐、易丢失、不便携的问题，以及提高个人信息的安全性和可靠性，我们结合了课程所学的理论知识和个人所思，设计了一个简单高效的个人信息管理系统，帮助用户更好的组织和管理各类个人信息。

本小组设计的个人信息管理系统基于C++，主要管理用户的个人信息、消费记录、成绩信息，包括：“添加并保存信息、查询信息、修改/删除各类信息”等基本功能。为了提高系统的安全性，本系统还包含了登录、注册功能，以此为用户信息进行加密处理。

本系统的亮点在于：

1. 在每一步操作完之后都会保持该系统的原界面，以便用户多次使用相同功能，节省时间；
2. 在每一个功能界面旁都有一个主菜单按钮，若用户按错功能键，可快速跳转至主菜单，无需重启管理系统；
3. 在有成绩信息、消费记录出现的界面均可直接增删改查对应信息，无需进行多余的操作，提高了用户工作效率。
4. 系统设计

本系统是以C++为基础，合理运用函数、结构体、指针、链表等知识，实现的模拟个人信息管理系统，用于高效管理学生的个人信息。

系统的主要功能模块如下：

* 登录/注册

用户进入主菜单前必须先登录个人账号，若用户名或密码不符，则无法进入用户主菜单，这一步骤大大提高了账号信息的安全性。

* 添加学生

此处用户可添加学生的个人信息。

* 查询学生

此处可根据学生学号或编号来查询学生个人信息、消费记录以及成绩信息，也可实现增删改查功能。

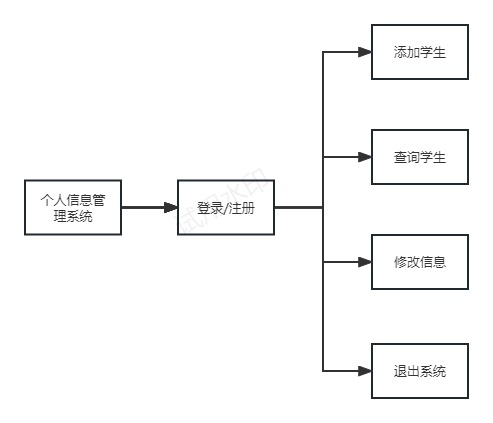
* 修改信息

此处可根据学生学号或编号来修学生个人信息、消费记录以及成绩信息，也可实现增删改查功能。

* 退出系统

另 ：上述每一步骤都可直接返回主菜单，无需重启程序即可一次实现多种功能。

功能结构图如下：

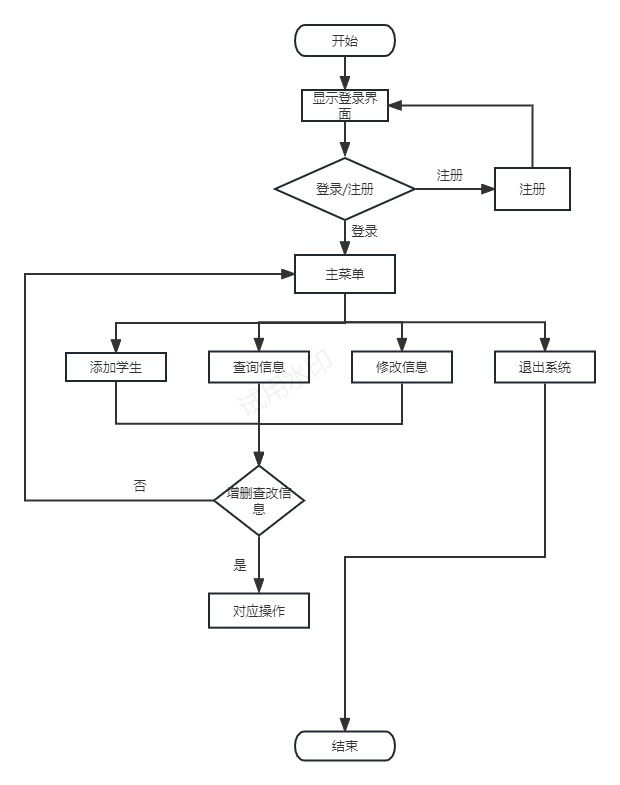


本组的分工合作安排：

刘军男同学作为技术主力，主要负责系统的编程工作；

黄虹霖同学作为辅助，主要负责结合课设系统的设计撰写结题报告、调试程序。

1. 系统功能图



1. 系统码源

account.h

#ifndef ACCOUNT\_H

#define ACCOUNT\_H

#include <QString>

class Account

{

private: QString user,password;

public:

Account();

Account(QString tuser, QString tpassword){

user = tuser;

password = tpassword;

}

QString getUser(){

return user;

}

QString getPassword(){

return password;

}

};

#endif // ACCOUNT\_H

billcore.h

#ifndef BILLCORE\_H

#define BILLCORE\_H

#include <QString>

class BillCore

{

private: QString id,date,item,charge;

public:

BillCore();

BillCore(QString tid,QString tdate,QString titem, QString tcharge){

id = tid;

date = tdate;

item = titem;

charge = tcharge;

}

QString getId(){

return id;

}

QString getDate(){

return date;

}

QString getItem(){

return item;

}

QString getCharge(){

return charge;

}

void setId(QString tid){

id = tid;

}

void setDate(QString tdate){

date = tdate;

}

void setItem(QString titem){

item = titem;

}

void setCharge(QString tcharge){

charge = tcharge;

}

};

#endif // BILLCORE\_H

billrecord.h

#ifndef BILLRECORD\_H

#define BILLRECORD\_H

#include <QWidget>

#include <QMessageBox>

#include <QtDebug>

#include <QFile>

#include <QDateEdit>

#include <QComboBox>

#include <QString>

namespace Ui {

class billrecord;

}

class billrecord : public QWidget

{

Q\_OBJECT

public:

explicit billrecord(QWidget \*parent = nullptr);

~billrecord();

private slots:

void on\_memu\_clicked();

void on\_btn\_add\_clicked();

void on\_btn\_search\_clicked();

void on\_btn\_change\_clicked();

void on\_btn\_delete\_clicked();

private:

Ui::billrecord \*ui;

};

#endif // BILLRECORD\_H

chan\_info.h

#ifndef CHAN\_INFO\_H

#define CHAN\_INFO\_H

#include <QWidget>

#include <QFile>

#include <QMessageBox>

#include <QtDebug>

#include <QVector>

#include "studentinfo.h"

namespace Ui {

class chan\_info;

}

class chan\_info : public QWidget

{

Q\_OBJECT

public:

explicit chan\_info(QWidget \*parent = nullptr);

~chan\_info();

private slots:

void on\_back\_clicked();

void on\_change\_clicked();

void on\_delete\_2\_clicked();

private:

Ui::chan\_info \*ui;

};

#endif // CHAN\_INFO\_H

Change.h

#ifndef CHANGE\_H

#define CHANGE\_H

#include <QWidget>

namespace Ui {

class change;

}

class change : public QWidget

{

Q\_OBJECT

public:

explicit change(QWidget \*parent = nullptr);

~change();

private slots:

void on\_back\_clicked();

void on\_stu\_info\_clicked();

void on\_consume\_clicked();

void on\_score\_clicked();

private:

Ui::change \*ui;

};

#endif // CHANGE\_H

functions.h

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

#include <QWidget>

namespace Ui {

class functions;

}

class functions : public QWidget

{

Q\_OBJECT

public:

explicit functions(QWidget \*parent = nullptr);

~functions();

private slots:

void on\_changeinf\_2\_clicked();

void on\_addstu\_clicked();

void on\_search\_clicked();

void on\_changeinf\_clicked();

private:

Ui::functions \*ui;

};

#endif // FUNCTIONS\_H

gpa.h

#ifndef GPA\_H

#define GPA\_H

#include <QString>

class GPA

{

private: QString name,id,subject,score;

public:

GPA();

GPA(QString tname, QString tid, QString tsubject, QString tscore){

name = tname;

id = tid;

subject = tsubject;

score = tscore;

}

QString getName(){

return name;

}

QString getId(){

return id;

}

QString getSubject(){

return subject;

}

QString getScore(){

return score;

}

void setId(QString tid){

id = tid;

}

void setName(QString tname){

name = tname;

}

void setSubject(QString tsubject){

subject = tsubject;

}

void setScore(QString tscore){

score = tscore;

}

};

#endif // GPA\_H

Mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "memu.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clicked();

void on\_linkstart\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

menu.h

#ifndef MEMU\_H

#define MEMU\_H

#include <QWidget>

#include <QtDebug>

#include <QString>

#include <QFile>

#include <QVector>

#include <QMessageBox>

namespace Ui {

class memu;

}

class memu : public QWidget

{

Q\_OBJECT

public:

explicit memu(QWidget \*parent = nullptr);

~memu();

private slots:

void on\_btn\_login\_clicked();

void on\_btn\_regis\_clicked();

void on\_checkBox\_clicked(bool checked);

private:

Ui::memu \*ui;

};

#endif // MEMU\_H

newstu.h

#ifndef NEWSTU\_H

#define NEWSTU\_H

#include <QMessageBox>

#include <QtDebug>

#include <QFile>

#include <QDateEdit>

#include <QComboBox>

#include <QWidget>

#include <QString>

namespace Ui {

class newstu;

}

class newstu : public QWidget

{

Q\_OBJECT

public:

explicit newstu(QWidget \*parent = nullptr);

~newstu();

private slots:

void on\_back\_clicked();

void on\_addnew\_clicked();

private:

Ui::newstu \*ui;

};

#endif // NEWSTU\_H

regist.h

#ifndef REGIST\_H

#define REGIST\_H

#include <QtDebug>

#include <QString>

#include <QFile>

#include <QWidget>

namespace Ui {

class regist;

}

class regist : public QWidget

{

Q\_OBJECT

public:

explicit regist(QWidget \*parent = nullptr);

~regist();

private slots:

void on\_regis\_clicked();

void on\_bcak\_clicked();

private:

Ui::regist \*ui;

};

#endif // REGIST\_H

sear\_info.h

#ifndef SEAR\_INFO\_H

#define SEAR\_INFO\_H

#include <QWidget>

#include <QtDebug>

#include <QFile>

#include <QVector>

#include <QMessageBox>

namespace Ui {

class sear\_info;

}

class sear\_info : public QWidget

{

Q\_OBJECT

public:

explicit sear\_info(QWidget \*parent = nullptr);

~sear\_info();

private slots:

void on\_btn\_search\_clicked();

void on\_back\_clicked();

private:

Ui::sear\_info \*ui;

};

#endif // SEAR\_INFO\_H

search.h

#ifndef SEARCH\_H

#define SEARCH\_H

#include <QWidget>

namespace Ui {

class search;

}

class search : public QWidget

{

Q\_OBJECT

public:

explicit search(QWidget \*parent = nullptr);

~search();

private slots:

void on\_per\_info\_clicked();

void on\_consume\_clicked();

void on\_score\_clicked();

void on\_back\_clicked();

private:

Ui::search \*ui;

};

#endif // SEARCH\_H

studentinfo.h

#ifndef STUDENTINFO\_H

#define STUDENTINFO\_H

#include <QString>

class StudentInfo

{

private:

QString name,code,hobby,birth,gender,major;

public:

StudentInfo();

StudentInfo(QString tname, QString tcode, QString thobby, QString tbirth, QString tgender, QString tmajor){

name = tname;

code = tcode;

gender = tgender;

major = tmajor;

birth = tbirth;

hobby = thobby;

}

QString getCode(){

return code;

}

QString getName(){

return name;

}

QString getGender(){

return gender;

}

QString getBirth(){

return birth;

}

QString getHobby(){

return hobby;

}

QString getMajor(){

return major;

}

void setCode(QString tcode){

code = tcode;

}

void setName(QString tname){

name = tname;

}

void setGender(QString tgender){

gender = tgender;

}

void setHobby(QString thobby){

hobby = thobby;

}

void setBirth(QString tbirth){

birth = tbirth;

}

void setMajor(QString tmajor){

major = tmajor;

}

};

#endif // STUDENTINFO\_H

supersocre.h

#ifndef SUPERSOCRE\_H

#define SUPERSOCRE\_H

#include <QWidget>

#include <QMessageBox>

#include <QtDebug>

#include <QFile>

#include <QDateEdit>

#include <QComboBox>

#include <QString>

namespace Ui {

class supersocre;

}

class supersocre : public QWidget

{

Q\_OBJECT

public:

explicit supersocre(QWidget \*parent = nullptr);

~supersocre();

private slots:

void on\_memu\_clicked();

void on\_btn\_add\_clicked();

void on\_pushButton\_clicked();

void on\_btn\_change\_clicked();

void on\_btn\_delete\_clicked();

private:

Ui::supersocre \*ui;

};

#endif // SUPERSOCRE\_H

account.cpp

#include "account.h"

Account::Account()

{

}

billcore.cpp

#include "billcore.h"

BillCore::BillCore()

{

}

billrecord.cpp

#include "billrecord.h"

#include "ui\_billrecord.h"

#include "functions.h"

#include "billcore.h"

billrecord::billrecord(QWidget \*parent) :

QWidget(parent),

ui(new Ui::billrecord)

{

ui->setupUi(this);

}

billrecord::~billrecord()

{

delete ui;

}

//返回主菜单

void billrecord::on\_memu\_clicked()

{

functions \*w = new functions;

w->show();

this->hide();

}

//添加消费记录

void billrecord::on\_btn\_add\_clicked()

{

QString id = ui->ad\_id->text();

QString item = ui->ad\_item->text();

QString charge = ui->ad\_charge->text();

QString date = ui->ad\_date->text();

if(ui->ad\_id->text() == "" || ui->ad\_item->text() == "" || ui->ad\_charge->text() == ""){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QFile file("bill.txt");

//实例化一个QFile file为我们的数据文件bill.txt

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

//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，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

//QTextStream可以进行一些基本的文本读写，比如QString int char之类的数据QDataStream可以进行一个如QMap QPoint之类数据的读写。

out << id << " " << date << " " << item << " " << charge << endl;

//将我们刚刚获取的数据写入文件

file.close();

QMessageBox::about(this, "反馈", "插入成功");

billrecord \*w = new billrecord;

w->show();

this->hide();

}

//查询消费记录

void billrecord::on\_btn\_search\_clicked()

{

if(ui->sear\_id->text() == ""){

QMessageBox::about(this, "警告", "编号不能为空");

return;

}

QFile file("bill.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<BillCore> BILL;

//数据类型为BillCore的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString id,date,item,charge;

inp >> id >> date >> item >> charge;

BILL.push\_back(BillCore(id,date,item,charge));

//调用之前建立的构造函数实例化一个BillCore对象并将其加入BILL

}

BILL.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString id = ui->sear\_id->text();

bool flag = false;

for(auto i : BILL){

if(i.getId() == id){

ui->s\_date->setText(i.getDate());

ui->s\_item->setText(i.getItem());

ui->s\_charge->setText(i.getCharge());

flag = true;

break;

}

}

if(!flag){

QMessageBox::about(this, "反馈", "编号不存在！");

}

ui->sear\_id->clear();

}

//修改消费记录

void billrecord::on\_btn\_change\_clicked()

{

if(ui->c\_id->text() == "" || ui->c\_item->text() == "" || ui->c\_charge->text() == ""){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QFile file("bill.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<BillCore> BILL;

//数据类型为BillCore的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString id,date,item,charge;

inp >> id >> date >> item >> charge;

BILL.push\_back(BillCore(id,date,item,charge));

//调用之前建立的构造函数实例化一个BillCore对象并将其加入BILL

}

BILL.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString id = ui->c\_id->text();

QString item = ui->c\_item->text();

QString charge = ui->c\_charge->text();

QString date = ui->c\_date->text();

bool flag = false;

for(QVector<BillCore>::iterator it = BILL.begin(); it != BILL.end(); it++){

if(it->getId() == id){

it->setDate(date);

it->setItem(item);

it->setCharge(charge);

flag = true;

}

}

if(flag){ //如果进行过修改，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "修改成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : BILL){

out << i.getId() << " " << i.getDate() << " " << i.getItem() << " " << i.getCharge() << endl;

}

file.close();

}else{

//如果没有进行修改，弹出不存在对话框

QMessageBox::about(this, "反馈", "编号不存在！");

}

//关闭文件

billrecord \*w = new billrecord;

w->show();

this->hide();

//初始化

}

//删除消费记录

void billrecord::on\_btn\_delete\_clicked()

{

QVector<BillCore> BILL;

if(ui->c\_id->text() == ""){

QMessageBox::about(this, "警告", "编号不能为空");

return;

}

QFile file("bill.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

while(!inp.atEnd()){ //读到文件结尾

QString id,date,item,charge;

inp >> id >> date >> item >> charge;

BILL.push\_back(BillCore(id,date,item,charge));

//调用之前建立的构造函数实例化一个BillCore对象并将其加入BILL

}

BILL.pop\_back();

//扔掉最后的无用数据

file.close();

QString id = ui->c\_id->text();

bool flag = false;

for(QVector<BillCore>::iterator it = BILL.begin(); it != BILL.end(); it++){

if(it->getId() == id){

BILL.erase(it);//迭代器遍历BILL找到对应就删除

flag = true;

}

}

if(flag){ //如果进行过修改，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "删除成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : BILL){

out << i.getId() << " " << i.getDate() << " " << i.getItem() << " " << i.getCharge() << endl;

}

file.close();

}else{

//如果没有进行删除，弹出不存在对话框

QMessageBox::about(this, "反馈", "编号不存在！");

}

ui->c\_id->clear();

}

chan\_info.cpp

#include "chan\_info.h"

#include "ui\_chan\_info.h"

#include "change.h"

chan\_info::chan\_info(QWidget \*parent) :

QWidget(parent),

ui(new Ui::chan\_info)

{

ui->setupUi(this);

}

chan\_info::~chan\_info()

{

delete ui;

}

//返回

void chan\_info::on\_back\_clicked()

{

change \*w = new change;

w->show();

this->hide();

}

//修改

void chan\_info::on\_change\_clicked()

{

if(ui->name->text() == "" || ui->code->text() == "" || ui->major->text() == ""||ui->hobby->text() == ""){

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QVector<StudentInfo> allStudentInfo; //数据类型为StudentInfo的QVector容器

QFile file("student.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

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

QTextStream inp(&file);

//以file建立一个QT的文本流

while(!inp.atEnd()){

QString name,code,major,gender,birth,hobby;

inp >> name >> code >> major >> gender >> birth >> hobby;

//读入各项信息

allStudentInfo.push\_back(StudentInfo(name, code, hobby, birth, gender, major));

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

}

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

file.close();

QString code = ui->code->text();

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

QString gender = ui->gender->currentText();

QString major = ui->major->text();

QString birth = ui->date->text();

QString hobby = ui->hobby->text();

bool flag = false;

for(QVector<StudentInfo>::iterator it = allStudentInfo.begin(); it != allStudentInfo.end(); it++){

if(it->getCode() == code){

it->setName(name);

it->setGender(gender);

it->setMajor(major);

it->setBirth(birth);

it->setHobby(hobby);

flag = true;

}

}

if(flag){ //如果进行过修改，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "修改成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : allStudentInfo){

out << i.getName() << " " << i.getCode() << " " << i.getMajor() << " " << i.getGender() << " " << i.getBirth() << " " << i.getHobby() << endl;

}

file.close();

}else{

//如果没有进行修改，弹出不存在对话框

QMessageBox::about(this, "反馈", "学号不存在！");

}

//关闭文件

chan\_info \*w = new chan\_info;

w->show();

this->hide();

//初始化

}

//删除

void chan\_info::on\_delete\_2\_clicked()

{

QVector<StudentInfo> allStudentInfo; //数据类型为StudentInfo的QVector容器

if(ui->code->text() == ""){ //如果code输入栏为空则输出错误提示并返回

QMessageBox::about(this, "反馈", "学号不得为空!");

return;

}

QFile file("student.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

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

QTextStream inp(&file);

//以file建立一个QT的文本流

while(!inp.atEnd()){

QString name,code,major,gender,birth,hobby;

inp >> name >> code >> major >> gender >> birth >> hobby;

//读入各项信息

allStudentInfo.push\_back(StudentInfo(name, code, hobby, birth, gender, major));

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

}

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

file.close();

QString code = ui->code->text();

//获取用户输入的学号

bool flag = false;

//记录是否进行过删除

for(QVector<StudentInfo>::iterator it = allStudentInfo.begin(); it != allStudentInfo.end(); it++){

//用迭代器遍历allStudentInfo

if(it->getCode() == code){ //如果找到有学号与输入学号相同的学生，就进行删除

allStudentInfo.erase(it);

flag = true;

}

}

if(flag){ //如果进行过删除，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "删除成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : allStudentInfo){

out << i.getName() << " " << i.getCode() << " " << i.getMajor() << " " << i.getGender() << " " << i.getBirth() << " " << i.getHobby() << endl;

}

file.close();

}else{

//如果没有进行删除，弹出不存在对话框

QMessageBox::about(this, "反馈", "学号不存在！");

}

//关闭文件

ui->code->clear();

//清空id输入框

}

change.cpp

#include "change.h"

#include "ui\_change.h"

#include "functions.h"

#include "chan\_info.h"

#include "billrecord.h"

#include "supersocre.h"

change::change(QWidget \*parent) :

QWidget(parent),

ui(new Ui::change)

{

ui->setupUi(this);

}

change::~change()

{

delete ui;

}

//返回主菜单

void change::on\_back\_clicked()

{

functions \*w = new functions;

w->show();

this->hide();

}

//个人信息修改

void change::on\_stu\_info\_clicked()

{

chan\_info \*w = new chan\_info;

w->show();

this->hide();

}

//消费记录修改

void change::on\_consume\_clicked()

{

billrecord \*w = new billrecord;

w->show();

this->hide();

}

//成绩修改

void change::on\_score\_clicked()

{

supersocre \*w = new supersocre;

w->show();

this->hide();

}

functions.cpp

#include "functions.h"

#include "ui\_functions.h"

#include "newstu.h"

#include "change.h"

#include "search.h"

functions::functions(QWidget \*parent) :

QWidget(parent),

ui(new Ui::functions)

{

ui->setupUi(this);

}

functions::~functions()

{

delete ui;

}

//（忘记重命名了www删干净头文件库太麻烦了）退出系统按钮

void functions::on\_changeinf\_2\_clicked()

{

exit(0);

}

//添加学生

void functions::on\_addstu\_clicked()

{

newstu \*w = new newstu();

w->show();

this->hide();

}

//查找学生

void functions::on\_search\_clicked()

{

search \*w = new search();

w->show();

this->hide();

}

//修改信息

void functions::on\_changeinf\_clicked()

{

change \*w = new change;

w->show();

this->hide();

}

gpa.cpp

#include "gpa.h"

GPA::GPA()

{

}

main.cpp

#include "mainwindow.h"

#include "memu.h"

#include "regist.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

memu w;

w.show();

return a.exec();

}

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "memu.h"

#include "functions.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

//退出系统

void MainWindow::on\_pushButton\_clicked()

{

exit(0);

}

//转到功能选择主菜单

void MainWindow::on\_linkstart\_clicked()

{

functions \*w = new functions();

w->show();

this->hide();

}

menu.cpp

#include "memu.h"

#include "mainwindow.h"

#include "ui\_memu.h"

#include "regist.h"

#include <QString>

#include <QMessageBox>

#include "account.h"

memu::memu(QWidget \*parent) :

QWidget(parent),

ui(new Ui::memu)

{

ui->setupUi(this);

}

memu::~memu()

{

delete ui;

}

//登录(查找账户和密码）

void memu::on\_btn\_login\_clicked()

{

//失败发送提示

if(ui->user->text() == ""||ui->password->text() == ""){

QMessageBox::about(this, "警告", "用户名或密码不能为空");

return;

}

QFile file("account.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<Account> acer;

//数据类型为Account的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString user,password;

inp >> user >> password;

acer.push\_back(Account(user, password));

//调用之前建立的构造函数实例化一个Account对象并将其加入acer

}

acer.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString user = ui->user->text();

QString password = ui->password->text();

bool flag = false;

for(auto i : acer){

if((i.getUser() == user) && (i.getPassword() == password))

{

flag = true;

//成功进入主界面

MainWindow \*w = new MainWindow();

w->show();

this->hide();

}

}

if(!flag){

QMessageBox::about(this, "反馈", "用户名或密码错误!");

}

ui->user->clear();

ui->password->clear();

}

//注册账户

void memu::on\_btn\_regis\_clicked()

{

regist \*w = new regist();

w->show();

this->hide();

}

//密码显示

void memu::on\_checkBox\_clicked(bool checked)

{

if(checked){

ui->password->setEchoMode(QLineEdit::Normal);

}//显示 Normal, NoEcho, Password, PasswordEchoOnEdit

else{

ui->password->setEchoMode(QLineEdit::Password);

}//密文

}

newstu.cpp

#include "newstu.h"

#include "ui\_newstu.h"

#include "functions.h"

newstu::newstu(QWidget \*parent) :

QWidget(parent),

ui(new Ui::newstu)

{

ui->setupUi(this);

}

newstu::~newstu()

{

delete ui;

}

//返回主菜单

void newstu::on\_back\_clicked()

{

functions \*w = new functions();

w->show();

this->hide();

}

//添加信息到库

void newstu::on\_addnew\_clicked()

{

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

//将用户输入的姓名由nameLineEdit取出

QString code = ui->code->text();

//将用户输入的学号由codeLineEdit取出

QString major = ui->major->text();

//将用户输入的专业由majorLineEdit取出

QString gender = ui->gender->currentText();

//将用户选择的信息从gender里面取出

QString date = ui->date->text();

// QString dateString = date.toString("yyyy-MM-dd");好像不用

//将dateEdit里的日期取出

QString hobby = ui->hobby->text();

//取出爱好

if(ui->name->text() == "" || ui->code->text() == "" || ui->major->text() == ""||ui->hobby->text() == "" ){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QFile file("student.txt");

//实例化一个QFile file为我们的数据文件student.txt

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

//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，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

//QTextStream可以进行一些基本的文本读写，比如QString int char之类的数据QDataStream可以进行一个如QMap QPoint之类数据的读写。

out << name << " " << code << " " << major << " " << gender <<" "<< date <<" "<< hobby << endl;

//将我们刚刚获取的数据写入文件

file.close();

QMessageBox::about(this, "反馈", "插入成功");

//直接开新窗口doge

newstu \*w = new newstu;

w->show();

this->close();

/\*ui->name->clear();

ui->code->clear();

ui->major->clear();

ui->hobby->clear();

//将用户输入的数据清空

直接开新窗口呗（天才！

//date和gender项暂时没有想到好的初始化方案（反正有选项先不管bushi\*/

}

regist.cpp

#include "regist.h"

#include "ui\_regist.h"

#include "memu.h"

#include <QString>

#include <QMessageBox>

regist::regist(QWidget \*parent) :

QWidget(parent),

ui(new Ui::regist)

{

ui->setupUi(this);

}

regist::~regist()

{

delete ui;

}

//注册用户信息

void regist::on\_regis\_clicked()

{

QString str1 = ui->passw1->text();

QString str2 = ui->passw2->text();

//判断两次密码是否相同

int i = QString::compare(str1,str2);

if(i){

QMessageBox::about(this,"警告","两次密码输入不同");

return;

}

//密码个数报错

QString str = ui->passw1->text();

if(str.length()<8){

QMessageBox::about(this,"警告","密码个数不符");

return;

}

QFile file("account.txt");

//实例化一个QFile file为我们的数据文件account.txt

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

//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，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

//QTextStream可以进行一些基本的文本读写，比如QString int char之类的数据QDataStream可以进行一个如QMap QPoint之类数据的读写。

QString account = ui->username->text();

//取出账号信息

out << account << " " << str1 << "\n";

//将我们刚刚获取的数据写入文件

file.close();

QMessageBox::about(this, "反馈", "注册成功");

//注册完成跳转到登录窗口

memu \*w = new memu();

w->show();

this->hide();

}

//返回登录界面

void regist::on\_bcak\_clicked()

{

memu \*w = new memu();

w->show();

this->hide();

}

sear\_info.cpp

#include "sear\_info.h"

#include "ui\_sear\_info.h"

#include "search.h"

#include "studentinfo.h"

sear\_info::sear\_info(QWidget \*parent) :

QWidget(parent),

ui(new Ui::sear\_info)

{

ui->setupUi(this);

}

sear\_info::~sear\_info()

{

delete ui;

}

//查询信息

void sear\_info::on\_btn\_search\_clicked()

{

if(ui->lineEdit->text() == ""){

QMessageBox::about(this, "警告", "学号不能为空");

return;

}

QFile file("student.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<StudentInfo> allStudentInfo;

//数据类型为StudentInfo的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString name, code, gender, major, birth, hobby;

inp >> name >> code >> major >> gender >> birth >> hobby;

allStudentInfo.push\_back(StudentInfo(name, code, hobby, birth, gender, major));

//allStudentInfo.push\_back(1);

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

}

allStudentInfo.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString code = ui->lineEdit->text();

bool flag = false;

for(auto i : allStudentInfo){

if(i.getCode() == code){

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

ui->showbirth->setText(i.getBirth());

ui->showgender->setText(i.getGender());

ui->showmajor->setText(i.getMajor());

ui->showhobby->setText(i.getHobby());

flag = true;

break;

}

}

if(!flag){

QMessageBox::about(this, "反馈", "学号不存在！");

}

ui->lineEdit->clear();

}

//返回查找界面

void sear\_info::on\_back\_clicked()

{

search \*w = new search;

w->show();

this->hide();

}

search.cpp

#include "search.h"

#include "ui\_search.h"

#include "functions.h"

#include "sear\_info.h"

#include "billrecord.h"

#include "supersocre.h"

search::search(QWidget \*parent) :

QWidget(parent),

ui(new Ui::search)

{

ui->setupUi(this);

}

search::~search()

{

delete ui;

}

//个人信息查询

void search::on\_per\_info\_clicked()

{

sear\_info \*w = new sear\_info;

w->show();

this->hide();

}

//消费记录查询

void search::on\_consume\_clicked()

{

billrecord \*w = new billrecord;

w->show();

this->hide();

}

//成绩查询

void search::on\_score\_clicked()

{

supersocre \*w = new supersocre;

w->show();

this->hide();

}

//返回

void search::on\_back\_clicked()

{

functions \*w = new functions;

w->show();

this->hide();

}

studentinfo.cpp

#include "studentinfo.h"

StudentInfo::StudentInfo()

{

}

supersocre.cpp

#include "supersocre.h"

#include "ui\_supersocre.h"

#include "functions.h"

#include "gpa.h"

//supersocre 拼错了但是删ui改.h太麻烦了（已经删过一次了因为score的重合的问题）

supersocre::supersocre(QWidget \*parent) :

QWidget(parent),

ui(new Ui::supersocre)

{

ui->setupUi(this);

}

supersocre::~supersocre()

{

delete ui;

}

//返回主菜单

void supersocre::on\_memu\_clicked()

{

functions \*w = new functions;

w->show();

this->hide();

}

//添加成绩

void supersocre::on\_btn\_add\_clicked()

{

QString name = ui->a\_name->text();

QString id = ui->a\_id->text();

QString subject = ui->a\_subject->text();

QString score = ui->a\_score->text();

if(ui->a\_name->text() == "" || ui->a\_id->text() == "" || ui->a\_subject->text() == "" || ui->a\_score->text() == ""){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QFile file("score.txt");

//实例化一个QFile file为我们的数据文件score.txt

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

//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，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

//QTextStream可以进行一些基本的文本读写，比如QString int char之类的数据QDataStream可以进行一个如QMap QPoint之类数据的读写。

out << name << " " << id << " " << subject << " " << score << endl;

//将我们刚刚获取的数据写入文件

file.close();

QMessageBox::about(this, "反馈", "插入成功");

supersocre \*w = new supersocre;

w->show();

this->hide();

}

//查询成绩

void supersocre::on\_pushButton\_clicked()

{

if(ui->s\_id->text() == "" || ui->s\_subject->text() == ""){

QMessageBox::about(this, "警告", "存在空项");

return;

}

QFile file("score.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<GPA> gpaw;

//数据类型为BillCore的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString name,id,subject,score;

inp >> name >> id >> subject >> score;

gpaw.push\_back(GPA(name,id,subject,score));

//调用之前建立的构造函数实例化一个GPA对象并将其加入gpaw

}

gpaw.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString id = ui->s\_id->text();

QString subject = ui->s\_subject->text();

bool flag = false;

for(auto i : gpaw){

if(i.getId() == id && i.getSubject() == subject){

ui->s\_name->setText(i.getName());

ui->s\_score->setText(i.getScore());

flag = true;

break;

}

}

if(!flag){

QMessageBox::about(this, "反馈", "成绩不存在！");

}

ui->s\_id->clear();

ui->s\_subject->clear();

}

//修改成绩

void supersocre::on\_btn\_change\_clicked()

{

if(ui->c\_name->text() == "" || ui->c\_id->text() == "" || ui->c\_subject->text() == "" || ui->c\_score->text() == ""){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "反馈", "存在空项");

return;

}

QFile file("score.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

QVector<GPA> gpaw;

//数据类型为BillCore的QVector容器

while(!inp.atEnd()){ //读到文件结尾

QString name,id,subject,score;

inp >> name >> id >> subject >> score;

gpaw.push\_back(GPA(name,id,subject,score));

//调用之前建立的构造函数实例化一个GPA对象并将其加入gpaw

}

gpaw.pop\_back();

//扔掉最后的无用数据

file.close();

//关闭文件

QString name = ui->c\_name->text();

QString id = ui->c\_id->text();

QString subject = ui->c\_subject->text();

QString score = ui->c\_score->text();

bool flag = false;

for(QVector<GPA>::iterator it = gpaw.begin(); it != gpaw.end(); it++){

if(it->getId() == id && it->getSubject() == subject){

it->setName(name);

it->setScore(score);

flag = true;

}

}

if(flag){ //如果进行过修改，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "修改成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : gpaw){

out << i.getName() << " " << i.getId() << " " << i.getSubject() << " " << i.getScore() << endl;

}

file.close();

}else{

//如果没有进行修改，弹出不存在对话框

QMessageBox::about(this, "反馈", "成绩不存在！");

}

//关闭文件

supersocre \*w = new supersocre;

w->show();

this->hide();

//初始化

}

//删除成绩

void supersocre::on\_btn\_delete\_clicked()

{

QVector<GPA> gpaw;

if(ui->c\_id->text() == "" || ui->c\_subject->text() == ""){

//插入项数据都不能为空，否则在读取文件时会出现问题。

QMessageBox::about(this, "警告", "学号和科目不得为空！ ");

return;

}

QFile file("score.txt");

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

//以只读的方式打开文本文件

if(!file.isOpen()){ //文件打开失败

QMessageBox::about(this, "反馈", "文件打开失败");

return;

}

QTextStream inp(&file);

//以file作为Qt文本流

while(!inp.atEnd()){ //读到文件结尾

QString name,id,subject,score;

inp >> name >> id >> subject >> score;

gpaw.push\_back(GPA(name,id,subject,score));

//调用之前建立的构造函数实例化一个GPA对象并将其加入gpaw

}

gpaw.pop\_back();

//扔掉最后的无用数据

file.close();

QString id = ui->c\_id->text();

QString subject = ui->c\_subject->text();

bool flag = false;

for(QVector<GPA>::iterator it = gpaw.begin(); it != gpaw.end(); it++){

if(it->getId() == id && it->getSubject() == subject){

gpaw.erase(it);//迭代器遍历gpaw找到对应就删除

flag = true;

}

}

if(flag){ //如果进行过修改，弹出对话框并更新文件

QMessageBox::about(this, "反馈", "删除成功");

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

//以只写覆盖的方式打开文本文件

if(!file.isOpen()){ //如果数据文件没有打开，弹出对话框提示用户

QMessageBox::about(this, "反馈", "数据文件打开失败");

return;

}

QTextStream out(&file);

for(auto i : gpaw){

out << i.getName() << " " << i.getId() << " " << i.getSubject() << " " << i.getScore() << endl;

}

file.close();

}else{

//如果没有进行删除，弹出不存在对话框

QMessageBox::about(this, "反馈", "成绩不存在！");

}

ui->c\_id->clear();

ui->c\_subject->clear();

}

1. 运行结果及分析

程序运行结果如下：

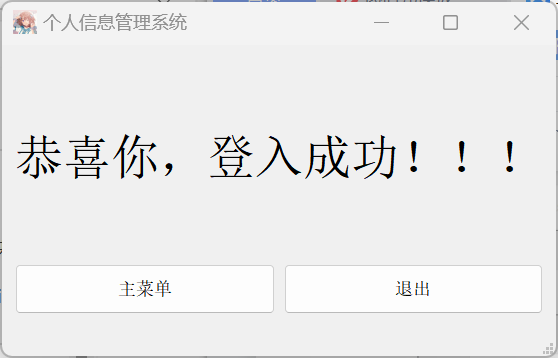
* 登录/注册



上图为登录界面，此时应输入正确的用户名及密码，点击“登录”，即可进入主菜单；若用户名与密码不符，则系统会给出相应提示。或点击注册，先注册账号再登录。



上图为注册页面，按照要求填写用户名及密码，点击“注册”，即可返回登录界面，或点击“返回”，也可进入登录界面。



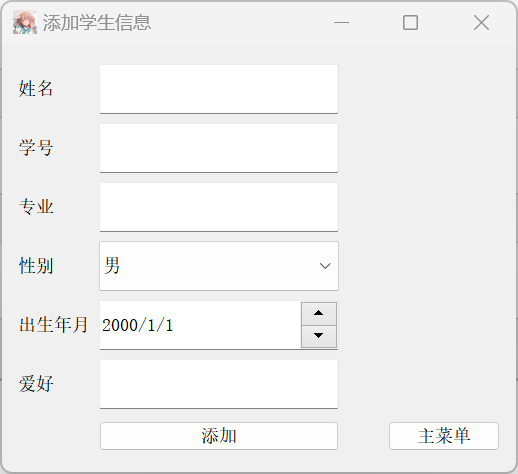
上图表示登录成功，点击“主菜单”，即可进入菜单系统；点击“退出”，则退出程序。

* 主菜单

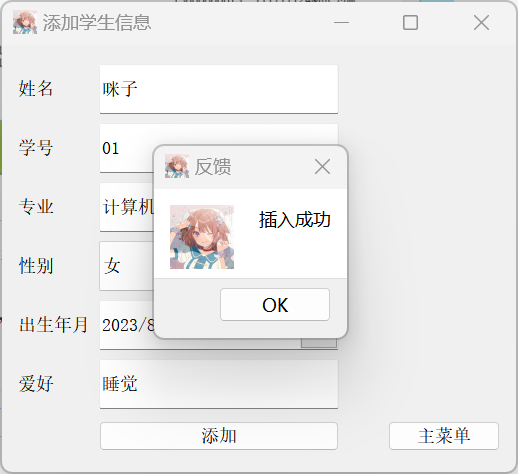


1. 添加学生

点击“添加学生”即可进入以下页面。

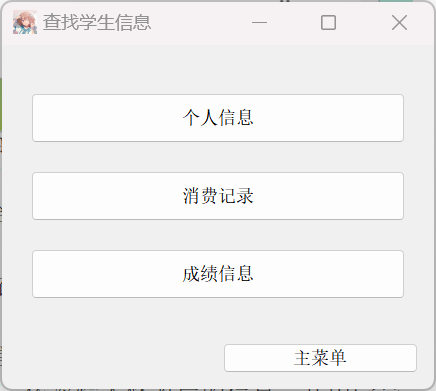


填写完整相应信息后，点击“添加”，出现以下提示时，表示信息添加成功。



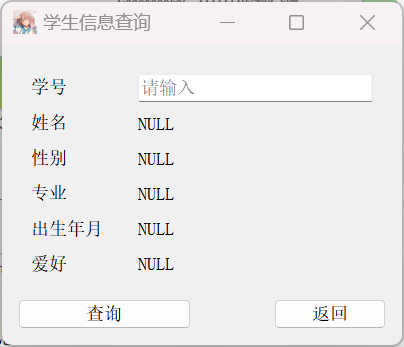
点击“OK”可继续添加学生信息，点击“主菜单”则跳转至菜单系统。

1. 查询学生



以上为“查询学生”页面。

1. 个人信息



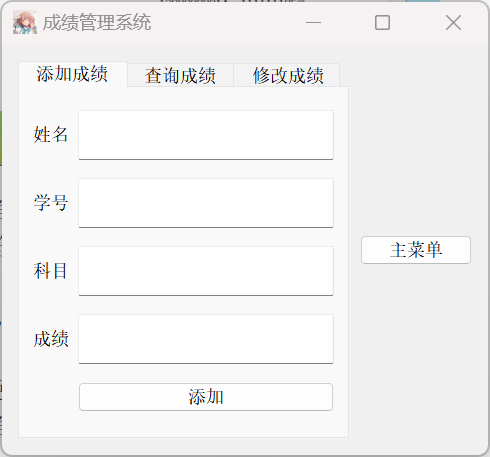
以上为查询学生信息页面，输入相应的学号，点击“查询”，即可查询到该生的基本信息；点击“返回”，则跳转至“查询学生”页面。

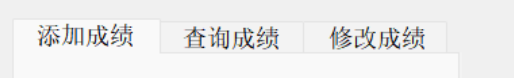
1. 消费记录



以上为“消费记录系统”页面，填写相应信息，可通过顶部按钮N(5Q4F}HO6[C7X85OJ57E5X实现添加、查询、修改/删除消费记录等功能。

1. 成绩信息



以上为“成绩管理系统”页面，填写相应信息，可通过顶部按钮实现添加、查询、修改/删除成绩等功能。

1. 修改信息



以上为“修改信息”页面。

1. 学生信息



填写相关信息，即可实现修改、删除学生信息功能；点击“返回”，则跳转至“修改信息”页面。

1. 消费记录
2. 成绩信息

（同（二）中的2. 3.）

**注意**：在使用任意功能时信息都要填写完整，不能存在空项，否则系统将会报错！

1. 设计总结

个人信息管理系统可以使用户更好地管理个人信息，提高工作和生活效率，无论是在工作还是日常生活中，个人信息管理系统都能为用户提供极大的便利和帮助。起初决定开始做这个系统时，大家都觉得这是一件轻而易举的事，但通过实践后发现这并不是一件易事。在编程过程中不断试错，在回忆旧知识的同时，学着使用新的开发工具与技术，在“0”和“1”的世界里，一次次尝试从“0”向“1”迈进。在这个过程中，技术主力花费了大量的时间和精力学习使用Qt、搭建环境、编写代码，自主学习能力和编程能力都有了质的飞跃，获得了极大的进步；辅助在这其中也更为深入的了解了一门新的编程语言，提高了论文撰写能力。当然，在整个项目完成的过程中，也存在着许多困难与挑战，比如说调试过程中无法添加信息、无法查找已添加过的信息等，我们通过自查代码、沟通、请教、查阅资料等方式逐一攻破难题。

经过这一次课设，我们受益匪浅，在团结协作的过程中，我们的沟通和交流能力大大提高，同时在学习的过程中，我们有了很好的机会自省，发现自己更多的不足，并发现了学习并非易事，需要长期的努力和坚持，学习也不能固步自封，要多与老师、同学交流，才能走得更远。在今后的学习中，我们必定会不断努力，以谦逊的姿态获取更大的进步。

参考文献

[1]木550.学生信息管理系统——C++版.CSDN.2023.[学生信息管理系统——C++版\_学生信息管理系统C++程序设计\_木550的博客-CSDN博客](https://blog.csdn.net/qq_63099085/article/details/126002741)

[2]非线性光学元件.C++图形化界面——含图形、按钮、鼠标、进度条等部件制作（带详细代码、讲解及注释）CSDN.2020.https://blog.csdn.net/weixin\_44044411/article/details/104276757