**1北京科技大学实验报告**

学院：机械工程学院 专业：机械系 班级：机械2308

姓名： 学号： 实验日期： 2024年 6月 4日

**实验名称：个人银行账户管理系统的改进**

**实验目的：**

完善银行账户系统，并实现以下功能：

(简述你的系统新增了哪些“银行1-6”没有的功能，每个功能用1句话简述)

1. 增加了日期构造异常处理机制

2. 增加了账户错误的相关异常处理

3. 增加了图形界面，登录与操作界面。

4. 图形界面有当月还款的提示和收入支出统计

5. 可以查询某个月按照时间排序的账户查询信息

6. 可以查询某个月按照交易金额从大到小排序的账户查询信息

**实验仪器：**

计算机：MECHREVO系列微机

CPU：12th Gen Intel(R) Core(TM) i9-12900HX，2300 Mhz，16 个内核，24 个逻辑处理器

内存：32GB

硬盘：1TB

显卡：NVIDIA GeForce RTX 4060 Laptop GPU

操作系统：Windows 11 家庭中文版

编译器：\_MSC\_VER 1939

集成开发环境：Microsoft Visual C++ 2022 17.9.4

**实验内容与步骤：**

【功能1：增加了日期构造异常处理机制】

1. 设计：定义了一个DateException用来处理日期的异常，当输入的日期不合法时，throw这个异常。
2. 实现

**class DateException : public exception {**

**public:**

**const char \*what() const noexcept override{**

**return "日期错误，请重新输入";**

**}**

**};**

**在date.cpp中调用**

**//日期构造或者输入错误**

**Date::Date(int year,int month,int day):year(year),month(month),day(day){**

**if ( day<=0 || day > getMaxDay()**

**|| month<=0 || month>12)**

**{**

**throw DateException();**

**}**

**};**

利用qt内置的日期

void MainWindow::on\_pushButton\_querymoney\_clicked()

{

querytype=1;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

void MainWindow::on\_pushButton\_querytime\_clicked()

{

querytype=0;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

3.测试

（1）测试用例1: 2008/12/30 2009/12/6 （用户输入正确的时间，因为直接用的qt里面内置的时间调节，所以q查询功能date并不会有异常，输入已经有限制了故就放一个测试示例）

运行截图如下：



**分析：**用户其实只是需要调整左右两个dateEdit内的时间，因为直接用的qt里面内置的时间调节，所以q查询功能date并不会有异常，输入已经有限制了故就放一个测试示例

【功能2：增加了账户错误的相关异常处理】

1. 设计

当增加的账户没有勾选（s或c）类型时抛出一个异常。

当存入或者取出的账户编号（id）不存在或账户下面的不同类型账户名重复时时抛出一个异常。（同时QMessage）

当从账户取钱时，余额不足，抛出一个异常（同时QMessage）。

当改变日期时，如果输入的日期小于今天或者超出本月最大日期，抛出异常（同时Qmessage弹窗）

sign一个总账户，重名，以及两次密码不一样（弹窗）

login一个账户，密码不正确

1. 实现

处理指令总体代码

引入了”iostream”与”QsqlDatabase”

**void MainWindow::process(char cmd){**

**date.show();**

**qDebug() << "\tTotal:" << Account::getTotal() << "\tcommand>";**

**switch (cmd) {**

**case 'a'://增加账户**

**try{**

**// cin >> type >> id;**

**type = account\_input[0].toStdString().c\_str()[0];**

**user = Username.toStdString();**

**id = account\_input[1].toStdString();//新账户名**

**qDebug()<<"ok3";**

**if(type == 's' || type == 'c'){**

**out << endl << cmd << " " << user << " ";**

**out << type << " " << id << " ";**

**}**

**else throw runtime\_error("账户类型错误");**

**}**

**catch (runtime\_error& err) {**

**//QMessageBox::warning(this,tr("警告"),tr(err.what());**

**qDebug()<<"a:runtime" << err.what() ;**

**}**

**if (type == 's') {**

**// 存入rate;**

**rate = account\_input[3].toDouble();**

**out<<rate;**

**account = new SavingsAccount(date, id, user, rate);**

**}**

**else {**

**// 存入credit信用额，rate，fee;**

**credit = account\_input[2].toDouble();**

**rate = account\_input[3].toDouble();**

**fee = account\_input[4].toDouble();**

**out<<credit<<" "<<rate<<" "<<fee;**

**account = new CreditAccount(date, id, user, credit, rate, fee);**

**qDebug()<<"ok4";**

**}**

**qDebug()<<"here\n";**

**koko->ui->textBrowser\_output->append(Username+ "的"+ QString::fromStdString(id)+ "账号已创建.");**

**qDebug()<<"ok5";**

**accounts.push\_back(account);**

**break;**

**case 'd'://存入现金**

**try{**

**// cin >> index\_ >> amount;由QString转换**

**index\_ = change\_input[0].toInt();**

**amount = change\_input[1].toDouble();**

**desc = change\_input[2].toStdString();**

**//deposit里其实已经检测过了，双重**

**if(index\_ < 0 || index\_ >= accounts.size()){**

**throw runtime\_error("The index you input is not exist.");**

**}**

**accounts[index\_]->deposit(date, amount, desc);**

**out << endl << cmd << " " ;**

**out<<index\_<<" "<<amount<<" ";**

**out<<desc;**

**show\_alert();**

**if(accounts[index\_]->getUser() == Username.toStdString()){ // 用户过滤**

**koko->ui->textBrowser\_output->append(QString::fromStdString(to\_string(date.getYear())+"年"+to\_string(date.getMonth())+"月"+to\_string(date.getDay())+"日"+accounts[index\_]->getId()+"存了"+to\_string(amount)+"元,原因："+desc));**

**}**

**}**

**catch (runtime\_error& err) {**

**qDebug()<<"d:runtime" << err.what();**

**}**

**break;**

**case 'w'://取出现金**

**try{**

**//cin >> index\_ >> amount;**

**index\_ = change\_input[0].toInt();**

**amount = change\_input[1].toDouble();**

**desc = change\_input[2].toStdString();**

**getline(cin, desc);**

**//QT中不存在抛出这个异常了前面deposit已经检测过了**

**if(index\_ < 0 || index\_ >= accounts.size()){**

**throw runtime\_error("The index you input is not exist.");**

**}**

**accounts[index\_]->withdraw(date, amount, desc);**

**out << endl << cmd << " ";**

**out<<index\_<<" "<<amount<<" ";**

**out<<desc;**

**show\_alert();**

**if(accounts[index\_]->getUser() == Username.toStdString()){ // 用户过滤**

**koko->ui->textBrowser\_output->append(QString::fromStdString(to\_string(date.getYear())+"年"+to\_string(date.getMonth())+"月"+to\_string(date.getDay())+"日"+accounts[index\_]->getId()+"取了"+to\_string(amount)+"元,原因："+desc));**

**}**

**}**

**//处理AccountException信息（异常）**

**catch (AccountException& err){//**

**qDebug() <<"w:AccountException" << err.what() ;**

**QMessageBox::warning(koko,tr("警告信息"),tr("没钱了"));**

**}**

**catch (runtime\_error& err) {**

**qDebug()<<"w:runtime\_err" << err.what() ;**

**}**

**break;**

**case 's'://查询各账户信息**

**static unsigned int num;**

**num = 1;**

**for (size\_t i = 0; i < accounts.size(); i++) {**

**//cout << "[" << i << "] ";**

**accounts[i]->show();**

**//cout << endl;**

**if(accounts[i]->getUser() == Username.toStdString()) // 用户过滤**

**if(accounts[i]->s\_c())**

**{**

**koko->ui->textBrowser\_output->append(QString::fromStdString("[" + to\_string(num++) + "] " +"类型:credit 账户名：" +accounts[i]->show()));**

**}**

**if(!accounts[i]->s\_c())**

**{**

**koko->ui->textBrowser\_output->append(QString::fromStdString("[" + to\_string(num++) + "] " +"类型:saving 账户名：" +accounts[i]->show()));**

**}**

**}**

**break;**

**case 'c'://改变日期**

**//cin >> day;**

**day = daychange;**

**if (day < date.getDay())**

**{**

**qDebug() << "You cannot specify a previous day";//日子已经过了**

**QMessageBox::warning(koko,tr("警告信息"),tr("听反方向的钟也回不去过去qwq"));**

**show\_alert();**

**break;**

**}**

**else if (day > date.getMaxDay() || day<1)**

**{**

**qDebug() << "Invalid day";**

**QMessageBox::warning(koko,tr("警告信息"),tr("无效日"));**

**show\_alert();**

**break;//终止，防止写入**

**}**

**else**

**date = Date(date.getYear(), date.getMonth(), day);**

**show\_alert();**

**out << endl << cmd << " ";**

**out<<day;**

**break;**

**case 'n'://进入下个月**

**if (date.getMonth() == 12)**

**date = Date(date.getYear() + 1, 1, 1);//下一年1月1月**

**else**

**date = Date(date.getYear(), date.getMonth() + 1, 1);//下一个月1号**

**for (vector<Account\*>::iterator iter = accounts.begin(); iter != accounts.end(); ++iter)**

**(\*iter)->settle(date);**

**show\_alert();**

**out << endl << cmd << " ";**

**break;**

**case 'q'://查询一段时间内的账目**

**try{**

**date1 = time1;**

**date2 = time2;**

**//Account::query(date1, date2 ,user);**

**koko->ui->textBrowser\_output->append(QString::fromStdString(Account::query(date1, date2, Username.toStdString(),querytype)));**

**}**

**//利用QT的日期所以其实并不会有这个DateException的异常（应该）**

**catch (DateException & err){**

**qDebug()<<"q:DataException" << err.what();**

**if(cin.fail()) cin.clear();**

**}**

**}**

**}**

**Qt槽函数创建账户**

**void MainWindow::on\_pushButton\_account\_clicked()**

**{**

**account\_input[1] = ui->lineEdit\_ID->text();**

**for(auto &iter : accounts){**

**if(iter->getId() == account\_input[1].toStdString()){**

**QMessageBox::warning(this,tr("警告信息"),tr("系统中已有重名账号"));**

**return;**

**}**

**}**

**if(ui->checkBox\_S->isChecked() && !ui->checkBox\_C->isChecked()){**

**account\_input[0]='s';**

**}**

**else if(!ui->checkBox\_S->isChecked() && ui->checkBox\_C->isChecked()){**

**account\_input[0]='c';**

**}**

**else{**

**QMessageBox::warning(this,tr("警告"),tr("请选择一种账户"));**

**return;**

**}**

**account\_input[2] = ui->lineEdit\_credit->text();**

**account\_input[3] = ui->lineEdit\_rate->text();**

**account\_input[4] = ui->lineEdit\_fee->text();**

**qDebug()<<"ok1";**

**MainWindow::process('a');**

**qDebug()<<"ok2";**

**}**

**Qt槽函数存钱**

void MainWindow::on\_pushButton\_deposit\_clicked()

{

change\_input[0] = ui->lineEdit\_DepositID->text();

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

if(accounts[i]->getId() == change\_input[0].toStdString()){

if(accounts[i]->getUser() != Username.toStdString()){

QMessageBox::warning(this,tr("警告"),tr("这不是你的账号,请不要乱动666"));

return;

}

change\_input[0] = (char)(i+48);

change\_input[1] = ui->lineEdit\_Depositamount->text();

change\_input[2] = ui->lineEdit\_Depositdesc->text();

MainWindow::process('d');

return;

}

}

QMessageBox::warning(this,tr("警告"),tr("找不到该账号haha"));

}

Qt槽函数取钱

void MainWindow::on\_pushButton\_withdraw\_clicked()

{

change\_input[0] = ui->lineEdit\_WithdrawID->text();

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

{//遍历所有对id

if(accounts[i]->getId() == change\_input[0].toStdString()){

if(accounts[i]->getUser() != Username.toStdString()){

QMessageBox::warning(this,tr("警告"),tr("这不是你的账号,请不要乱动666"));

return;

}

change\_input[0] = (char)(i+48);

change\_input[1] = ui->lineEdit\_Withdrawamount->text();

change\_input[2] = ui->lineEdit\_Withdrawdesc->text();

MainWindow::process('w');

return;

}

}

QMessageBox::warning(this,tr("警告"),tr("找不到该账号haha"));

}

改变日期

void MainWindow::on\_pushButton\_Confirmdays\_clicked()

{

koko->ui->textBrowser\_nowdate->clear();

daychange = koko->ui->lineEditday->text().toInt();

if(daychange || (daychange==0))

MainWindow::process('c');

else

QMessageBox::warning(this,tr("警告"),tr("需输入天数"));

//输出当前日期

/\*QString datenowdays = QString("当前日期 年: %1, 月: %2, 日: %3")

.arg(date.getYear())

.arg(date.getMonth())

.arg(date.getDay());

koko->ui->textBrowser\_nowdate->append(datenowdays);

\*/

}

Sign与login

//注册

void login::on\_Sign\_clicked()

{

QString username = this->ui->lineEdit\_loginusername->text();

QString passwordfirst = this->ui->lineEdit\_loginpassword1->text();

QString passwordsecond = this->ui->lineEdit\_loginpassword2->text();

// 检查用户名是否存在

QString checkCmd = QString("SELECT COUNT(\*) FROM userinfo WHERE username='%1'").arg(username);

QSqlQuery checkQuery;

if (checkQuery.exec(checkCmd))

{

checkQuery.next();

int count = checkQuery.value(0).toInt();//用户名计数，查询结果中的第一个字段，满足条件的行数，转为整数

if (count > 0)

{

qDebug() << "用户名已存在" << endl;

// 提示

QMessageBox::warning(this, "注册失败", "用户名已存在，请选择其他用户名。");

return;

}

}

else

{

qDebug() << "检查用户名失败" << endl;

return;

}

// 插入新用户信息

if(passwordfirst==passwordsecond)

{

QString insertCmd = QString("INSERT INTO userinfo (username, password) VALUES ('%1', '%2')").arg(username).arg(passwordfirst);

QSqlQuery insertQuery;

if (insertQuery.exec(insertCmd))

{

qDebug() << "插入成功" << endl;

QMessageBox::information(this, "注册成功", "账号创建成功！");

}

else

{

qDebug() << "插入失败" << endl;

QMessageBox::warning(this, "注册失败", "账号创建失败，请重试。");

}

}

else

{

QMessageBox::warning(this, "注册失败", "两次密码不一样，请重试。");

}

}

//注册

void login::on\_Login\_clicked()

{

QString username = this->ui->lineEdit\_username->text();

QString password = this->ui->lineEdit\_password->text();

//创建model

QSqlTableModel \*model = new QSqlTableModel;

//绑定表

model->setTable("userinfo");//main.cpp中已有说明

//过滤数据

model->setFilter(QString("username='%1' and password='%2'").arg(username).arg(password));

//查询

model->select();

//结果

int row = model->rowCount();

Username = username;

s = new MainWindow;

if(row>0){

qDebug()<<"登录成功"<<endl;

//打开主界面

s->show();

}

else{

QMessageBox::about(NULL,"","用户名或密码错误");

s->deleteLater();

qDebug()<<"登录失败"<<endl;

}

}

1. 测试

（1） 测试用例1：创建账户：ceshi 0.002 60 1000 未勾选类型（在qt界面输入）

输出截图：



分析：没有勾选类型，所以异常抛出

（2） 测试用例2：创建账户：ceshi 0.002 60 1000 勾选信用账户（在qt界面输入）

输出截图：



分析：左下角提示账户创建成功

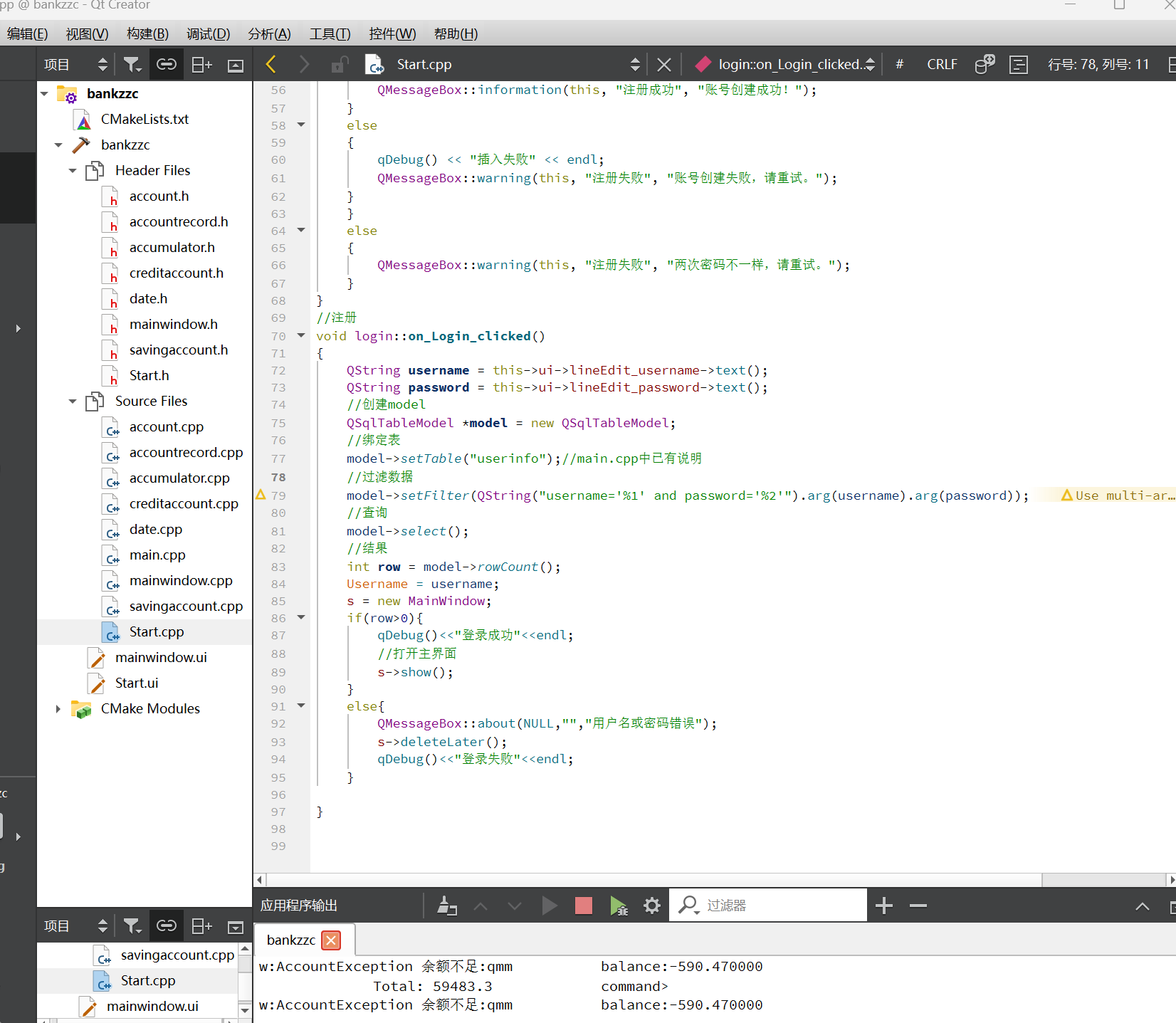
（3） 测试用例3：创建账户：qmm 0.002 60 1000 勾选信用账户（在qt界面输入重名的账户）



分析：因为创建了已有此名字的账户

（4） 测试用例4：取钱：qmm 5000（在qt界面输入取钱下面框的信息）

输出截图：

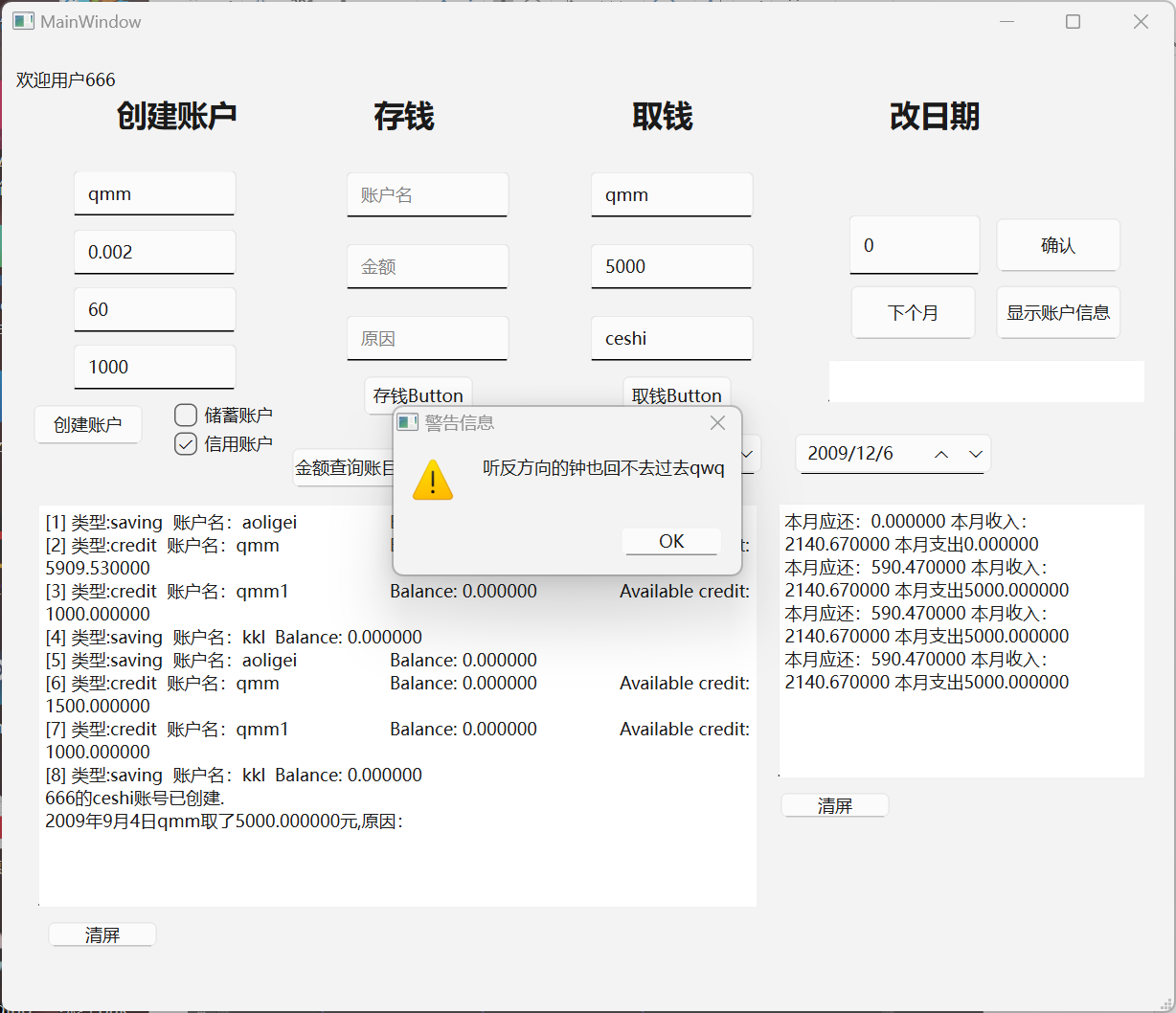


分析：取的钱超过余额和信用卡的额度了，所以便会弹窗同时Qt界面里面qDebug()也输出了

（5） 测试用例5：改日期： 31（在qt界面输入两次并点击确认）

改日期： 0

输出截图：



分析：9月没有31天，且（0）改日期不能回到过去（异常处理的代码在上面已展示）

（6） 测试用例6：注册：账号：666 两次密码：666（用户在登录界面注册已有的账户）

输出截图：



分析：账号名已注册过

（6） 测试用例7：登录：账号：666 密码：667（用户在登录界面登录）

输出截图：



分析：密码错误

【功能3：增加了图形页面】

1. 设计

采用qt6.7.1设计了图形页面，并且完成了相关的按钮，注册/登陆功能。

以下为主页面样式，包含了注册与登录分别的账户和密码的输入框和注册登陆的按钮。



之后是登录成功的操作界面，左上角有欢迎语，以及一些功能的实现框，信息的输出框（textbrowser）在相应框中输入内容点击相应的按钮后便会输出信息



2.实现

（mainwindow.h）

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPushButton>

#include <QLineEdit>

#include <QLabel>

#include <QMessageBox>

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\_qingping2\_clicked();

void on\_qingping1\_clicked();

//void on\_pushButton\_query\_clicked();

void on\_pushButton\_account\_clicked();

void on\_pushButton\_deposit\_clicked();

void on\_pushButton\_withdraw\_clicked();

void on\_pushButton\_showaccount\_clicked();

void on\_pushButton\_NextMonth\_clicked();

void on\_pushButton\_Confirmdays\_clicked();

void on\_pushButton\_querymoney\_clicked();

void on\_pushButton\_querytime\_clicked();

private:

Ui::MainWindow \*ui;

static void process(char cmd);//处理命令

static void read\_cmd();//读取命令

static MainWindow \*koko;

static void show\_alert();

};

#endif // MAINWINDOW\_H

(Start.h)

#ifndef START\_H

#define START\_H

#include "mainwindow.h"

#include <QWidget>

namespace Ui {

class login;

}

class login : public QWidget

{

Q\_OBJECT

public:

explicit login(QWidget \*parent = nullptr);

~login();

private slots:

void on\_Sign\_clicked();

void on\_Login\_clicked();

private:

Ui::login \*ui;

MainWindow \*s;

};

#endif // START\_H

(main.cpp)

#include "Start.h"

#include <QApplication>

#include <QString>

#include <QSqlQuery>

#include <QDebug>

#include <QSqlDatabase>

#define endl '\n'

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

{

// 创建QApplication实例

QApplication a(argc, argv);

// 创建数据库

QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");

// 设置数据库名字

db.setDatabaseName("login.db");

// 打开数据库

if(db.open())

{

qDebug() << "已打开数据库" << endl;

}

else

{

qDebug() << "未打开数据库" << endl;

}

// 创建一个存储username和password的的数据库表,表名userinfo已有则不创建

QString cmd = "create table if not exists userinfo(username varchar(64),password varchar(64))";

// 执行sql语句

QSqlQuery query;

if(query.exec(cmd))

{

qDebug() << "execute success" << endl;

}

else

{

qDebug() << "execute failure" << endl;

}

login login11;

// 打开登录界面

login11.show();

return a.exec();

}

(Start.cpp)

#include "Start.h"

#include "ui\_Start.h"

#include "mainwindow.h"

#include <QMovie>

#include <QSqlQuery>

#include <QDebug>

#include <QSqlTableModel>

#define endl '\n'

extern QString Username;//登录时的用户名

login::login(QWidget \*parent) :

QWidget(parent),

ui(new Ui::login)

{

ui->setupUi(this);

}

login::~login()

{

delete ui;

}

//注册

void login::on\_Sign\_clicked()

{

QString username = this->ui->lineEdit\_loginusername->text();

QString passwordfirst = this->ui->lineEdit\_loginpassword1->text();

QString passwordsecond = this->ui->lineEdit\_loginpassword2->text();

// 检查用户名是否存在

QString checkCmd = QString("SELECT COUNT(\*) FROM userinfo WHERE username='%1'").arg(username);

QSqlQuery checkQuery;

if (checkQuery.exec(checkCmd))

{

checkQuery.next();

int count = checkQuery.value(0).toInt();//用户名计数，查询结果中的第一个字段，满足条件的行数，转为整数

if (count > 0)

{

qDebug() << "用户名已存在" << endl;

// 提示

QMessageBox::warning(this, "注册失败", "用户名已存在，请选择其他用户名。");

return;

}

}

else

{

qDebug() << "检查用户名失败" << endl;

return;

}

// 插入新用户信息

if(passwordfirst==passwordsecond)

{

QString insertCmd = QString("INSERT INTO userinfo (username, password) VALUES ('%1', '%2')").arg(username).arg(passwordfirst);

QSqlQuery insertQuery;

if (insertQuery.exec(insertCmd))

{

qDebug() << "插入成功" << endl;

QMessageBox::information(this, "注册成功", "账号创建成功！");

}

else

{

qDebug() << "插入失败" << endl;

QMessageBox::warning(this, "注册失败", "账号创建失败，请重试。");

}

}

else

{

QMessageBox::warning(this, "注册失败", "两次密码不一样，请重试。");

}

}

//注册

void login::on\_Login\_clicked()

{

QString username = this->ui->lineEdit\_username->text();

QString password = this->ui->lineEdit\_password->text();

//创建model

QSqlTableModel \*model = new QSqlTableModel;

//绑定表

model->setTable("userinfo");//main.cpp中已有说明

//过滤数据

model->setFilter(QString("username='%1' and password='%2'").arg(username).arg(password));

//查询

model->select();

//结果

int row = model->rowCount();

Username = username;

s = new MainWindow;

if(row>0){

qDebug()<<"登录成功"<<endl;

//打开主界面

s->show();

}

else{

QMessageBox::about(NULL,"","用户名或密码错误");

s->deleteLater();

qDebug()<<"登录失败"<<endl;

}

}

(mainwindow.cpp)

#include "mainwindow.h"

#include "./ui\_mainwindow.h"

#include "date.h"

#include "account.h"

#include "savingaccount.h"

#include "creditaccount.h"

#include <iostream>

#include <vector>

#include <fstream>

#include <stdexcept>

#include <QLabel>

#include <QPushButton>

#include <QLineEdit>

#include <QDebug>

#include <QMessageBox>

#include <QSqlDatabase>

QString Username;

Date date(2008,11,1);//起始日期

vector<Account \*> accounts;//创建动态账户数组，元素个数为0

char cmd,type;

int index\_, day;

double amount, credit, rate, fee;

string user, id, desc;

Account\* account;

Date date1, date2;

ofstream out;//用于添加新命令

QString account\_input[5];//存放account信息，c类type,id,credit,rate,fee s类type,id, ,rate

QString change\_input[3];//存放存取钱的账户名，金额，原因

int daychange;

Date time1 ,time2;

int querytype;

MainWindow \*MainWindow::koko = nullptr;

// 文件存储格式

// User cmd ...

//读取命令

void MainWindow::read\_cmd(){

ifstream in;

in.open("commands.txt");//程序启动后，逐条读出commands.txt文件中的命令，并按命令对账户进行不同的操作，将各个账户恢复到上次退出程序时的状态

if(!in.is\_open()){

qDebug()<<"Cant find file";

return;

}

while(in>>cmd){

switch (cmd) {

case 'a'://增加账户

in >> user >> type >> id;

if (type == 's') {

in >> rate;

account = new SavingsAccount(date, id, user ,rate);

}

else {

in >> credit >> rate >> fee;

account = new CreditAccount(date, id, user, credit, rate, fee);

}

accounts.push\_back(account);

break;

case 'd'://存入现金

in >> index\_ >> amount;

getline(in,desc);

accounts[index\_]->deposit(date, amount, desc);

break;

case 'w'://取出现金

in >> index\_ >> amount ;

getline(in,desc);

accounts[index\_]->withdraw(date, amount, desc);

break;

case 'c'://改变日期

in >> day;

date = Date(date.getYear(), date.getMonth(), day);

break;

case 'n'://进入下个月

if (date.getMonth() == 12)

date = Date(date.getYear() + 1, 1, 1);//下一年1月1月

else

date = Date(date.getYear(), date.getMonth() + 1, 1);//下一个月1号

for (vector<Account\*>::iterator iter = accounts.begin(); iter != accounts.end(); ++iter)//遍历所有账户

(\*iter)->settle(date);//每月都有，但1月1才计算

break;

}

}

in.close();

}

struct deleter {

template <class T> void operator () (T\* p) { delete p; }

};

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

koko = this;//另外设置的新内部地址，ui是内部的所以是必需的

//初始语

QString theHello = "欢迎用户"+ Username;

ui->label\_chushi->setText(theHello);

//读取文件

read\_cmd();

//显示需处理信息，还款的提示，当月账户的统计

show\_alert();

out.open("commands.txt",ios::app);//开始处理新命令并把新命令添加到commands.txt文件中

}

MainWindow::~MainWindow()

{

delete ui;

}

//处理命令

void MainWindow::process(char cmd){

date.show();

qDebug() << "\tTotal:" << Account::getTotal() << "\tcommand>";

switch (cmd) {

case 'a'://增加账户

try{

// cin >> type >> id;

type = account\_input[0].toStdString().c\_str()[0];

user = Username.toStdString();

id = account\_input[1].toStdString();//新账户名

qDebug()<<"ok3";

if(type == 's' || type == 'c'){

out << endl << cmd << " " << user << " ";

out << type << " " << id << " ";

}

else throw runtime\_error("账户类型错误");

}

catch (runtime\_error& err) {

//QMessageBox::warning(this,tr("警告"),tr(err.what());

qDebug()<<"a:runtime" << err.what() ;

}

if (type == 's') {

// 存入rate;

rate = account\_input[3].toDouble();

out<<rate;

account = new SavingsAccount(date, id, user, rate);

}

else {

// 存入credit信用额，rate，fee;

credit = account\_input[2].toDouble();

rate = account\_input[3].toDouble();

fee = account\_input[4].toDouble();

out<<credit<<" "<<rate<<" "<<fee;

account = new CreditAccount(date, id, user, credit, rate, fee);

qDebug()<<"ok4";

}

qDebug()<<"here\n";

koko->ui->textBrowser\_output->append(Username+ "的"+ QString::fromStdString(id)+ "账号已创建.");

qDebug()<<"ok5";

accounts.push\_back(account);

break;

case 'd'://存入现金

try{

// cin >> index\_ >> amount;由QString转换

index\_ = change\_input[0].toInt();

amount = change\_input[1].toDouble();

desc = change\_input[2].toStdString();

//deposit里其实已经检测过了，双重

if(index\_ < 0 || index\_ >= accounts.size()){

throw runtime\_error("The index you input is not exist.");

}

accounts[index\_]->deposit(date, amount, desc);

out << endl << cmd << " " ;

out<<index\_<<" "<<amount<<" ";

out<<desc;

show\_alert();

if(accounts[index\_]->getUser() == Username.toStdString()){ // 用户过滤

koko->ui->textBrowser\_output->append(QString::fromStdString(to\_string(date.getYear())+"年"+to\_string(date.getMonth())+"月"+to\_string(date.getDay())+"日"+accounts[index\_]->getId()+"存了"+to\_string(amount)+"元,原因："+desc));

}

}

catch (runtime\_error& err) {

qDebug()<<"d:runtime" << err.what();

}

break;

case 'w'://取出现金

try{

//cin >> index\_ >> amount;

index\_ = change\_input[0].toInt();

amount = change\_input[1].toDouble();

desc = change\_input[2].toStdString();

getline(cin, desc);

//QT中不存在抛出这个异常了前面deposit已经检测过了

if(index\_ < 0 || index\_ >= accounts.size()){

throw runtime\_error("The index you input is not exist.");

}

accounts[index\_]->withdraw(date, amount, desc);

out << endl << cmd << " ";

out<<index\_<<" "<<amount<<" ";

out<<desc;

show\_alert();

if(accounts[index\_]->getUser() == Username.toStdString()){ // 用户过滤

koko->ui->textBrowser\_output->append(QString::fromStdString(to\_string(date.getYear())+"年"+to\_string(date.getMonth())+"月"+to\_string(date.getDay())+"日"+accounts[index\_]->getId()+"取了"+to\_string(amount)+"元,原因："+desc));

}

}

//处理AccountException信息（异常）

catch (AccountException& err){//

qDebug() <<"w:AccountException" << err.what() ;

QMessageBox::warning(koko,tr("警告信息"),tr("没钱了"));

}

catch (runtime\_error& err) {

qDebug()<<"w:runtime\_err" << err.what() ;

}

break;

case 's'://查询各账户信息

static unsigned int num;

num = 1;

for (size\_t i = 0; i < accounts.size(); i++) {

//cout << "[" << i << "] ";

accounts[i]->show();

//cout << endl;

if(accounts[i]->getUser() == Username.toStdString()) // 用户过滤

if(accounts[i]->s\_c())

{

koko->ui->textBrowser\_output->append(QString::fromStdString("[" + to\_string(num++) + "] " +"类型:credit 账户名：" +accounts[i]->show()));

}

if(!accounts[i]->s\_c())

{

koko->ui->textBrowser\_output->append(QString::fromStdString("[" + to\_string(num++) + "] " +"类型:saving 账户名：" +accounts[i]->show()));

}

}

break;

case 'c'://改变日期

//cin >> day;

day = daychange;

if (day < date.getDay())

{

qDebug() << "You cannot specify a previous day";//日子已经过了

QMessageBox::warning(koko,tr("警告信息"),tr("听反方向的钟也回不去过去qwq"));

show\_alert();

break;

}

else if (day > date.getMaxDay() || day<1)

{

qDebug() << "Invalid day";

QMessageBox::warning(koko,tr("警告信息"),tr("无效日"));

show\_alert();

break;//终止，防止写入

}

else

date = Date(date.getYear(), date.getMonth(), day);

show\_alert();

out << endl << cmd << " ";

out<<day;

break;

case 'n'://进入下个月

if (date.getMonth() == 12)

date = Date(date.getYear() + 1, 1, 1);//下一年1月1月

else

date = Date(date.getYear(), date.getMonth() + 1, 1);//下一个月1号

for (vector<Account\*>::iterator iter = accounts.begin(); iter != accounts.end(); ++iter)

(\*iter)->settle(date);

show\_alert();

out << endl << cmd << " ";

break;

case 'q'://查询一段时间内的账目

try{

date1 = time1;

date2 = time2;

//Account::query(date1, date2 ,user);

koko->ui->textBrowser\_output->append(QString::fromStdString(Account::query(date1, date2, Username.toStdString(),querytype)));

}

//利用QT的日期所以其实并不会有这个DateException的异常（应该）

catch (DateException & err){

qDebug()<<"q:DataException" << err.what();

if(cin.fail()) cin.clear();

}

}

}

void MainWindow::show\_alert(){

double debt = 0;

double income = 0, cost = 0;

double debting;

QString datenowdays = QString("当前日期 年: %1, 月: %2, 日: %3")

.arg(date.getYear())

.arg(date.getMonth())

.arg(date.getDay());

koko->ui->textBrowser\_nowdate->setText("");

koko->ui->textBrowser\_nowdate->append(datenowdays);

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

if(accounts[i]->getUser() == Username.toStdString()){

if(accounts[i]->s\_c())

{//指回来类型信用卡c 1

debting=accounts[i]->get\_debt();

if(debting<0)

{

debt -= accounts[i]->get\_debt();

}

}

}

}

//cout << date.getYear()<<date.getMonth()-1<<1

income = Account::month\_income(Date(date.getYear(),date.getMonth(),1),Date(date.getYear(),date.getMonth(),date.getMaxDay()),Username.toStdString());//收入

cost = Account::month\_cost(Date(date.getYear(),date.getMonth(),1),Date(date.getYear(),date.getMonth(),date.getMaxDay()),Username.toStdString());//支出

koko->ui->textBrowser\_account->append(QString::fromStdString("本月应还："+to\_string(debt)+" 本月收入："+to\_string(income)+" 本月支出"+to\_string(cost)));

}

void MainWindow::on\_qingping2\_clicked()

{

koko->ui->textBrowser\_output->setText("");

}

void MainWindow::on\_qingping1\_clicked()

{

koko->ui->textBrowser\_account->setText("");

}

/\*

void MainWindow::on\_pushButton\_query\_clicked()

{

querytype=0;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

\*/

void MainWindow::on\_pushButton\_account\_clicked()

{

account\_input[1] = ui->lineEdit\_ID->text();

for(auto &iter : accounts){

if(iter->getId() == account\_input[1].toStdString()){

QMessageBox::warning(this,tr("警告信息"),tr("系统中已有重名账号"));

return;

}

}

if(ui->checkBox\_S->isChecked() && !ui->checkBox\_C->isChecked()){

account\_input[0]='s';

}

else if(!ui->checkBox\_S->isChecked() && ui->checkBox\_C->isChecked()){

account\_input[0]='c';

}

else{

QMessageBox::warning(this,tr("警告"),tr("请选择一种账户"));

return;

}

account\_input[2] = ui->lineEdit\_credit->text();

account\_input[3] = ui->lineEdit\_rate->text();

account\_input[4] = ui->lineEdit\_fee->text();

qDebug()<<"ok1";

MainWindow::process('a');

qDebug()<<"ok2";

}

void MainWindow::on\_pushButton\_deposit\_clicked()

{

change\_input[0] = ui->lineEdit\_DepositID->text();

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

if(accounts[i]->getId() == change\_input[0].toStdString()){

if(accounts[i]->getUser() != Username.toStdString()){

QMessageBox::warning(this,tr("警告"),tr("这不是你的账号,请不要乱动666"));

return;

}

change\_input[0] = (char)(i+48);

change\_input[1] = ui->lineEdit\_Depositamount->text();

change\_input[2] = ui->lineEdit\_Depositdesc->text();

MainWindow::process('d');

return;

}

}

QMessageBox::warning(this,tr("警告"),tr("找不到该账号haha"));

}

void MainWindow::on\_pushButton\_withdraw\_clicked()

{

change\_input[0] = ui->lineEdit\_WithdrawID->text();

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

{//遍历所有对id

if(accounts[i]->getId() == change\_input[0].toStdString()){

if(accounts[i]->getUser() != Username.toStdString()){

QMessageBox::warning(this,tr("警告"),tr("这不是你的账号,请不要乱动666"));

return;

}

change\_input[0] = (char)(i+48);

change\_input[1] = ui->lineEdit\_Withdrawamount->text();

change\_input[2] = ui->lineEdit\_Withdrawdesc->text();

MainWindow::process('w');

return;

}

}

QMessageBox::warning(this,tr("警告"),tr("找不到该账号haha"));

}

void MainWindow::on\_pushButton\_showaccount\_clicked()

{

MainWindow::process('s');

}

void MainWindow::on\_pushButton\_NextMonth\_clicked()

{

MainWindow::process('n');

}

void MainWindow::on\_pushButton\_Confirmdays\_clicked()

{

koko->ui->textBrowser\_nowdate->clear();

daychange = koko->ui->lineEditday->text().toInt();

if(daychange || (daychange==0))

MainWindow::process('c');

else

QMessageBox::warning(this,tr("警告"),tr("需输入天数"));

//输出当前日期

/\*QString datenowdays = QString("当前日期 年: %1, 月: %2, 日: %3")

.arg(date.getYear())

.arg(date.getMonth())

.arg(date.getDay());

koko->ui->textBrowser\_nowdate->append(datenowdays);

\*/

}

void MainWindow::on\_pushButton\_querymoney\_clicked()

{

querytype=1;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

void MainWindow::on\_pushButton\_querytime\_clicked()

{

querytype=0;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

3.测试

（1）测试用例1:登录界面注册以及登录用例已经在功能2中展示过了

（登录账户：666密码：666）

运行截图：



分析：登录成功

（2）测试用例2：点击“显示账户信息”（之前已经操作过所以有数据保存着）

运行截图：



分析：如图片左下角框，显示信息成功

（3）测试用例3：（创建信用类用户）

账户名：gangyuan 利率：0.0002 年费：66 信用额：10000

运行截图：



分析：创建成功，框内有文本提示

（3）测试用例3：（存钱）

账户名：gangyuan 金额：6666 原因：ok

运行截图：



分析：如左下角，存钱成功

（4）测试用例4：（取钱）

账户名：qmm 金额：100 原因：nook

运行截图：



分析

如右下角框中成功了

（5）测试用例5：（超额取钱）

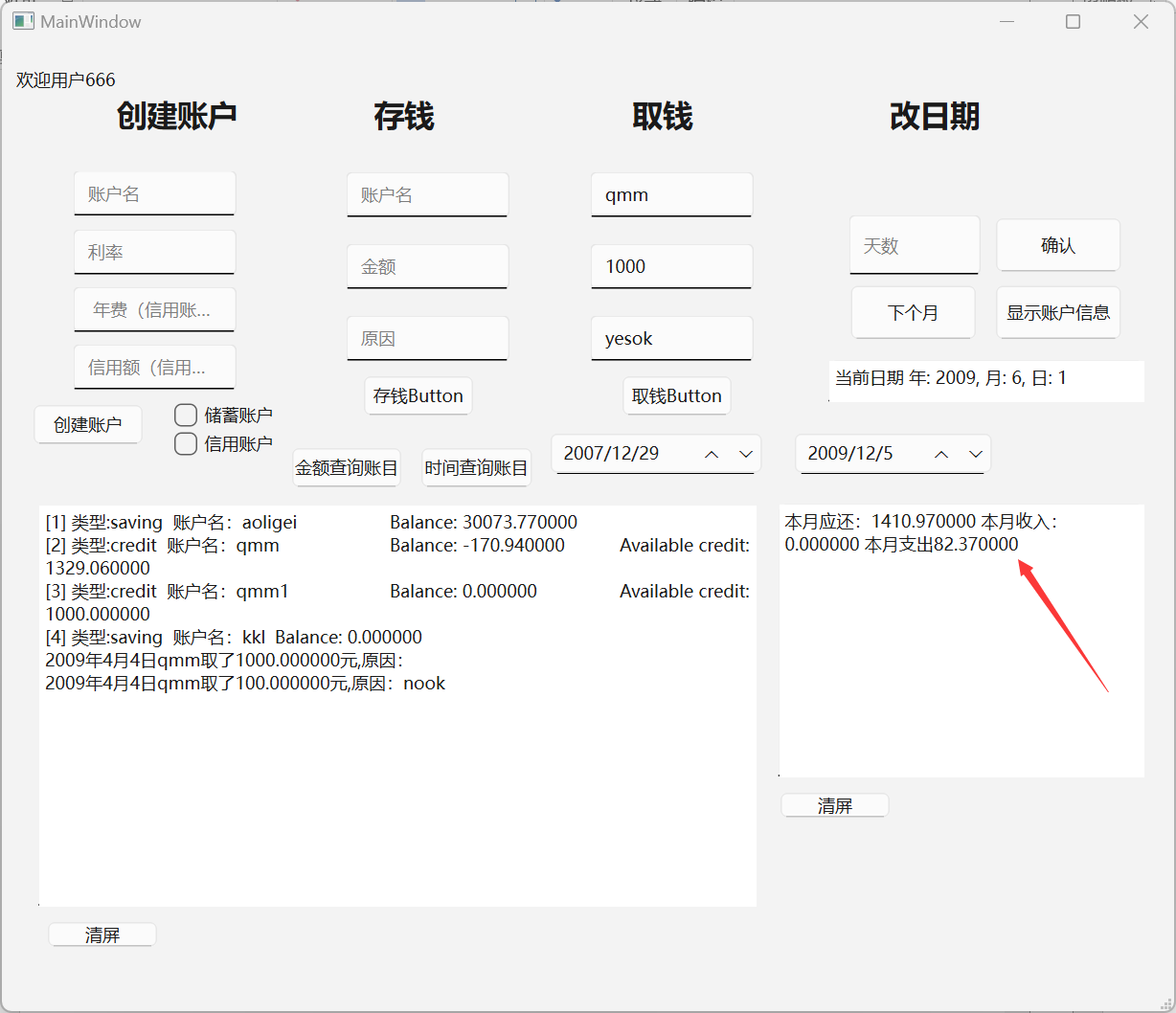
账户名：qmm 金额：1000 原因：yesok



分析：超额取钱了，抛出异常

（6）测试用例6：（点击进入下个月）

输出截图：



分析：右框有新输出

【功能4：图形界面有当月还款的提示和收入支出统计】

1. 设计

在右侧框显示本月支出和本月收入，用一个函数来统计，并且调用qt的方法来显示在文本框上。

2.实现

void MainWindow::show\_alert(){

double debt = 0;

double income = 0, cost = 0;

double debting;

QString datenowdays = QString("当前日期 年: %1, 月: %2, 日: %3")

.arg(date.getYear())

.arg(date.getMonth())

.arg(date.getDay());

koko->ui->textBrowser\_nowdate->setText("");

koko->ui->textBrowser\_nowdate->append(datenowdays);

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

if(accounts[i]->getUser() == Username.toStdString()){

if(accounts[i]->s\_c())

{//指回来类型信用卡c 1

debting=accounts[i]->get\_debt();

if(debting<0)

{

debt -= accounts[i]->get\_debt();

}

}

}

}

//cout << date.getYear()<<date.getMonth()-1<<1

income = Account::month\_income(Date(date.getYear(),date.getMonth(),1),Date(date.getYear(),date.getMonth(),date.getMaxDay()),Username.toStdString());//收入

cost = Account::month\_cost(Date(date.getYear(),date.getMonth(),1),Date(date.getYear(),date.getMonth(),date.getMaxDay()),Username.toStdString());//支出

koko->ui->textBrowser\_account->append(QString::fromStdString("本月应还："+to\_string(debt)+" 本月收入："+to\_string(income)+" 本月支出"+to\_string(cost)));

}

double Account::month\_cost(Date day1, Date day2, string user){

double cost = 0;

for(auto & it : recordMap){

if(day1 <= it.first && day2 >= it.first && it.second.account->getUser() == user){

if(it.second.amount < 0) cost -= it.second.amount;

}

}

return cost;

}

//本月收入 统计

double Account::month\_income(Date day1, Date day2, string user){

double incomes = 0;

for(auto & it : recordMap){

if(day1 <= it.first && day2 >=it.first && it.second.account->getUser() == user){

if(it.second.amount > 0) incomes += it.second.amount;

}

}

return incomes;

}

3.测试



分析：进行天数的改动都会执行此代码在右框输出信息

【功能5：可以查询某个月按照时间排序的账户查询信息】

1. 设计：在bank1-6其实已有实现时间输入为从qt提供的时间选择控件输入，然后点击时间查询账目，即可。

2.实现

void MainWindow::on\_pushButton\_querytime\_clicked()

{

querytype=0;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

3.测试：查询从2007/12/29到2009/12/25

输出截图：



分析：成功按时间查询

【功能6：可以查询某个月按照交易金额排序的账户查询信息】

1.设计：与银行1～6的类似，但是时间输入改为了从qt提供的时间选择控件输入，然后点击金额查询账目，即可。

2.实现

void MainWindow::on\_pushButton\_querymoney\_clicked()

{

querytype=1;

time1 = Date(ui->dateEdit1->date().year(), ui->dateEdit1->date().month(), ui->dateEdit1->date().day());

time2 = Date(ui->dateEdit2->date().year(), ui->dateEdit2->date().month(), ui->dateEdit2->date().day());

MainWindow::process('q');

}

3.测试：查询从2007/12/29到2009/12/25

输出截图：



分析：成功按交易金额查询

**实验结论与感想**

这个qt真是太有意思了，意外发现竟然原来是可以导入iostream和QSqlDatabase，锻炼我的C语言编写能力以及MySQL语句能力，同时以及文件的读写保存系统的运行也有一定了解（commands.txt）同时还有什么STL模块也觉得收益匪浅能和数据库后端等等结合起来非常不错，同时自我学习能力也得到提高，在编程计算机语言这块也更加了解。