**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：物联网工程 班级：物联24x

姓名：Char1esss 学号：U2024xxxxxx 实验日期：2025年6月11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.添加了图形界面

2.添加了月度统计功能

3.添加了账户错误异常处理

4.添加了查询账单功能

**实验仪器：**

计算机：Razer Blade 14 2024

CPU：AMD Ryzen 9 7940HS w

内存：32GB 5600MHz DDR5

硬盘：2TB

显卡：Radeon 780M Graphics/NVIDIA RTX 4070 Laptop

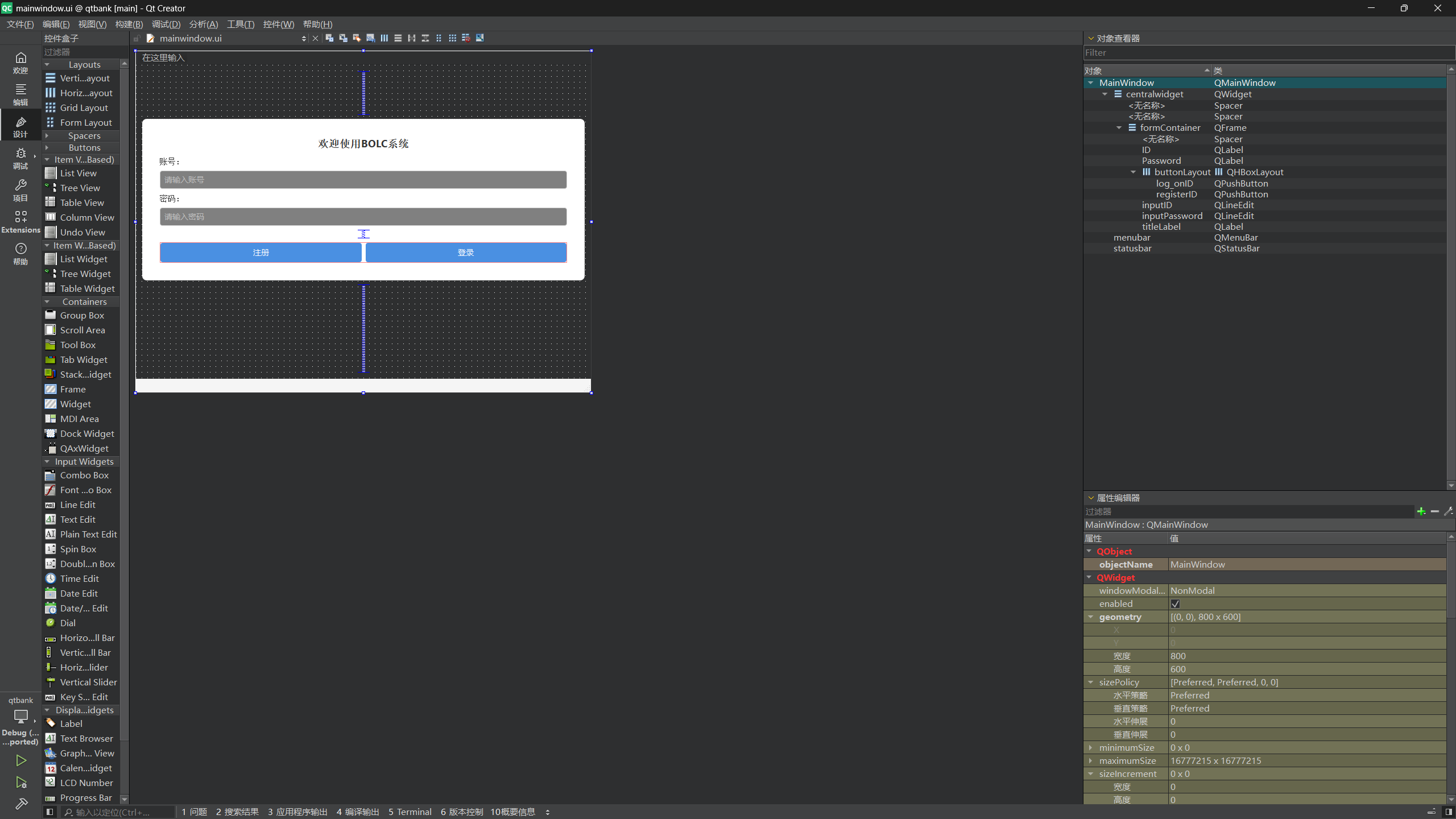
操作系统：Windows 11

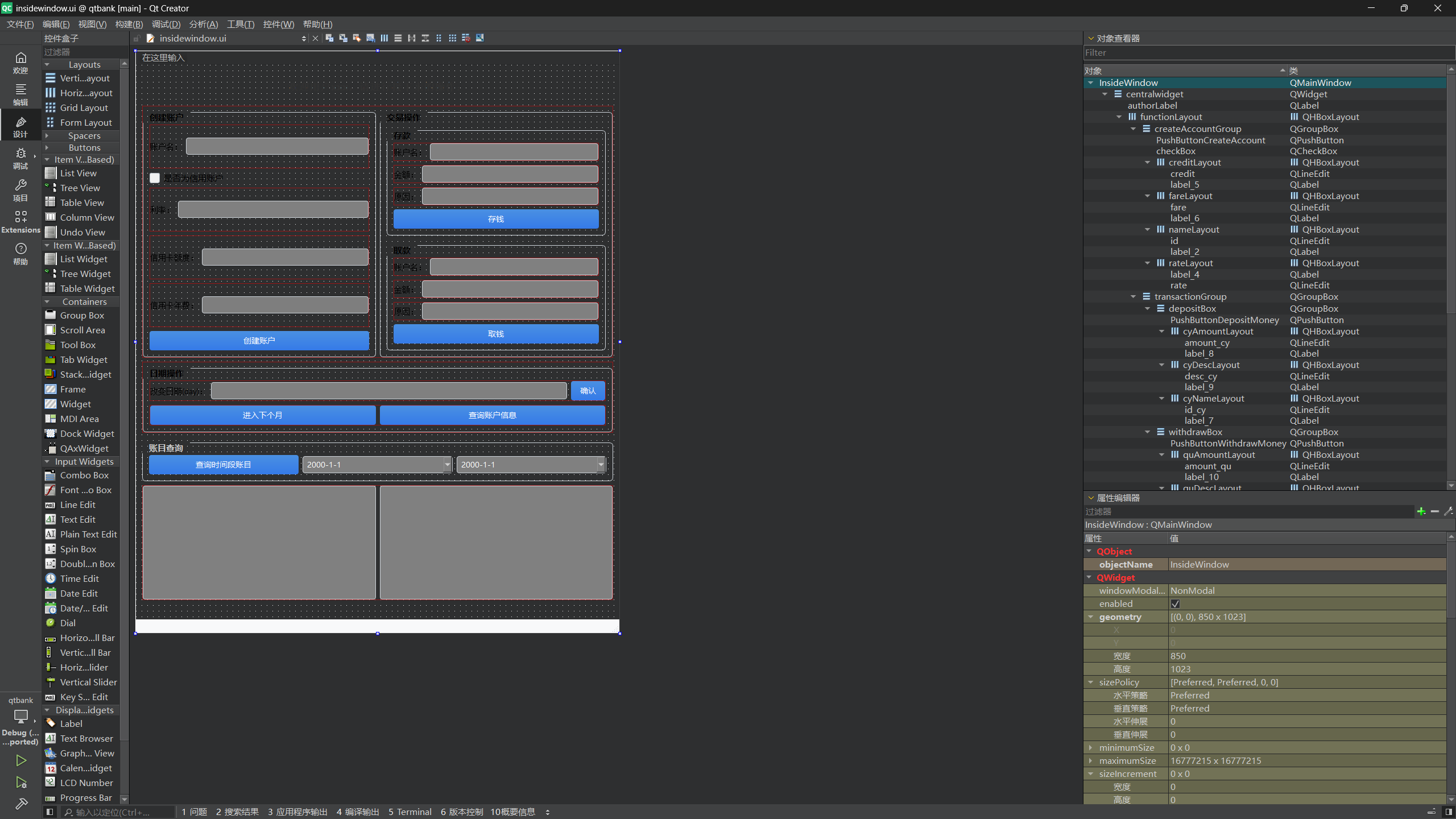
编译器：MinGW\_x86\_64

**实验内容与步骤：**

**【**功能1：添加了图形界面**】**

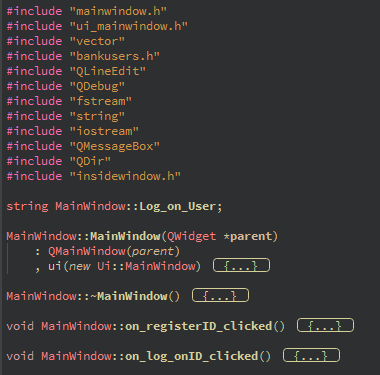
1. 设计

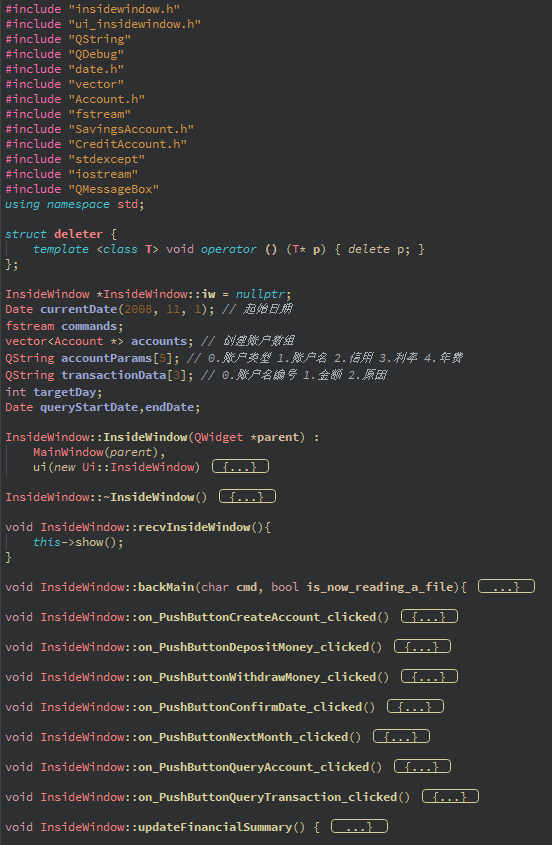
****



使用Qt设计了图形界面，通过输入栏和按钮等代替纯指令输入，实现注册登录和账户操作等功能，极大降低了操作难度。

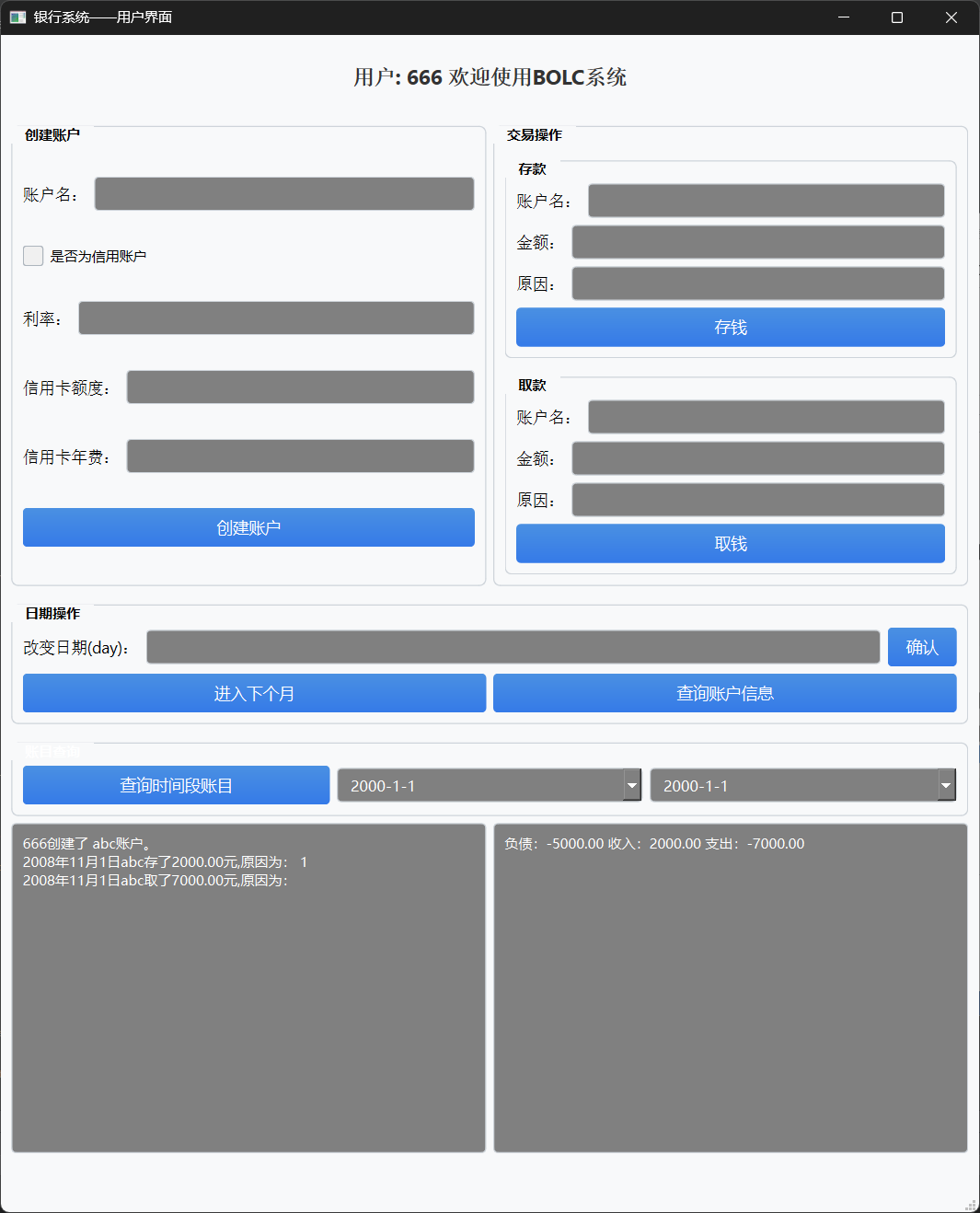
2.实现





代码片段1 图形界面相关代码实现（具体代码请看讲解视频或源代码）

3.测试

**** 

**【**功能2：添加了月度统计功能**】**

1. 设计：

添加了updateFinancialSummary()函数，在每次存款、取款等操作时更新相应数据，并在窗口中显示。

2.实现：

void InsideWindow::updateFinancialSummary() {

double totalDebt = 0, totalIncome = 0, totalExpense = 0;

// 获取当前月份的起止日期

Date startDate(currentDate.getYear(), currentDate.getMonth(), 1);

Date endDate(currentDate.getYear(), currentDate.getMonth(), currentDate.getMaxDay());

// 统计信用卡债务

for (Account\* account : accounts) {

if (account->getUser() == MainWindow::Log\_on\_User && account->isCreditAccount()) {

totalDebt = account->getDebt();

}

}

// 计算收入和支出

totalIncome = Account::calculateTotalIncomeBetweenDates(startDate, endDate, MainWindow::Log\_on\_User);

totalExpense = Account::calculateTotalExpenseBetweenDates(startDate, endDate, MainWindow::Log\_on\_User);

// 更新 UI 显示

iw->ui->textBrowser2->clear();

iw->ui->textBrowser2->append(QString("负债：%1 收入：%2 支出：%3")

.arg(QString::number(totalDebt, 'f', 2))

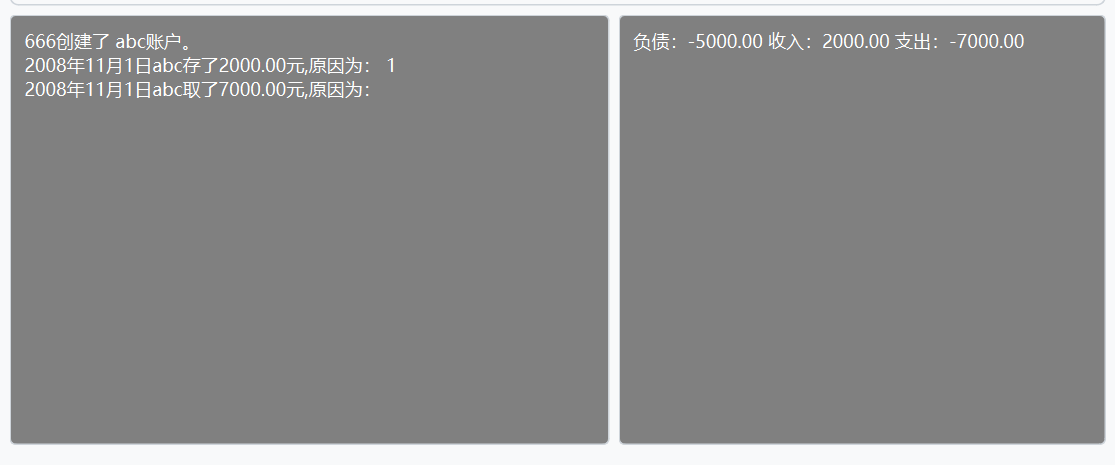
.arg(QString::number(totalIncome, 'f', 2))

.arg(QString::number(totalExpense, 'f', 2))

);

}

3.测试



**【**功能3：添加了账户错误异常处理**】**

1. 设计：

加的账户类型不是一个合法的参数时抛出异常。

当存/取款账户编号存在时抛出异常。

当存/取款金额非法时抛出异常。

当取款额度不足时抛出异常

1. 实现：

if (type == 's') {…}

else if (type == 'c') {…}

else throw runtime\_error("Not a saving or credit account.");

账户类型异常检测实现

if(index >= accounts.size()){

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

}

账户编号异常检测实现

void SavingsAccount::deposit(Date date, double amount, std::string desc) {

if (amount <= 0) {

throw std::runtime\_error("Amount should be positive");

}

record(date, amount, desc);

acc.change(date, getBalance());

}

void SavingsAccount::withdraw(Date date, double amount, std::string desc) {

if (amount <= 0) {

throw std::runtime\_error("Amount should be positive");

}

if(amount <= getBalance()) {

record(date, -amount, desc);

acc.change(date, getBalance());

}

else throw AccountException(this);

}

存/取款金额非法和余额不足异常检测

1. 测试
   1. 样例：command.txt内容如下，测试账户类型异常：

a z 666 abc 100 2 100

 结果：

* 1. 样例：command.txt内容如下，测试账户编号异常：

a c 666 abc 1000 2 100

d 1 1000 test

结果：

* 1. 样例：command.txt内容如下，测试存/取款金额异常：

a c 666 abc 1000 2 100

d 0 -1 test

结果：



3.4 样例：command.txt内容如下，测试存/取款余额不足异常：

a c 666 abc 1000 2 100

w 0 1100 test

结果：

**【**功能4：添加了查询账单功能**】**

1. 设计：

通过起始日期调用查询函数查找账户的交易记录，并输出在文本框。

2.实现

case 'q': // 查询一段时间内的账目

try {

date1 = queryStartDate;

date2 = endDate;

iw->ui->textBrowser1->append(QString::fromStdString(Account::query(date1, date2, MainWindow::Log\_on\_User)));

}

catch (DateException& err){

cerr << err.what() << endl;

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

}

break;

3. 测试：

测试样例：

结果：



**实验结论与感想**

非常好的实验，使我的大脑旋转。