**实验二**

**可视化窗口界面编程初探**

Windows **标准控件的使用**

**【实验目的】**

1． 了解 MFC 消息循环机制；

2． 掌握 MFC 中部分常用控件的使用方法；

3． 基于 MFC 编写具备一定用户界面交互功能的应用程序。

**【实验任务】**

1.

进一步理解 MFC 消息循环机制；

2.

进一步了解掌握 MFC 控件的使用方法，使用多种控件开发具有简单用户交互功能的

MFC 应用程序—个人信息简表；

3.

制作具有基本的数据处理能力的 MFC 应用程序—计算器。

**【实验设备】**

个人 PC，Windows11，Qt5.12。

**【计划学时】**

4-8 学时

**【实验内容】**

**1.具有简单用户界面交互功能的 MFC 应用程序，显示个人信息（采用Qt实现）**

功能要求

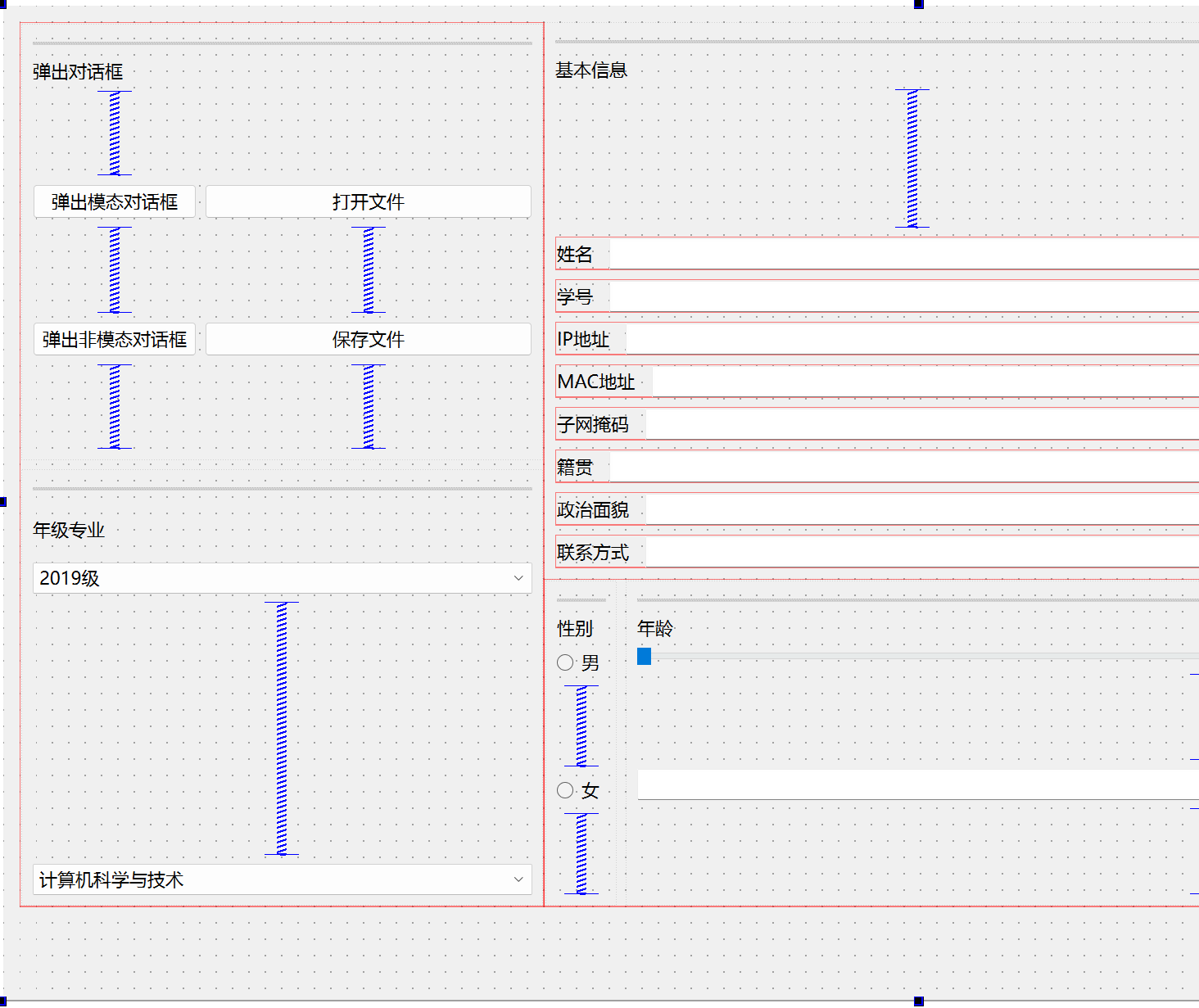
1. “弹出模态对话框”按钮无效(该控件不使能);

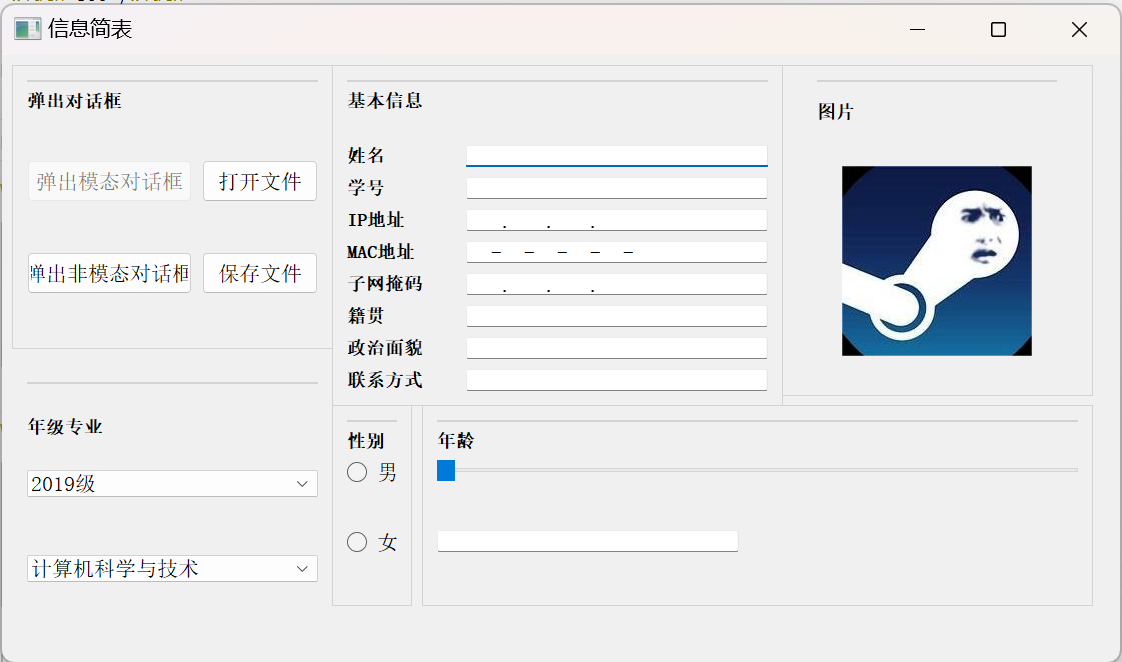
2. 点击“弹出非模态对话框”按钮，弹出一个你设计好的非模态对话框。

3. 点击“打开文件”按钮，弹出打开文件对话框，选中你上次保存的 txt 文件，点击“确定”后打开该文件，从中读取你的姓名，学号、和本地网络信息，动态填入对应的编辑框中。

开发工具：QtCreator；Qt版本：5.12；语言：C++14

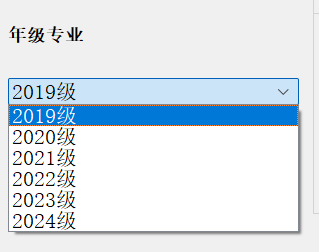
Ui界面：

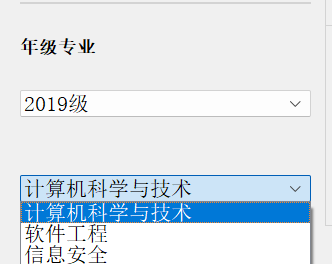




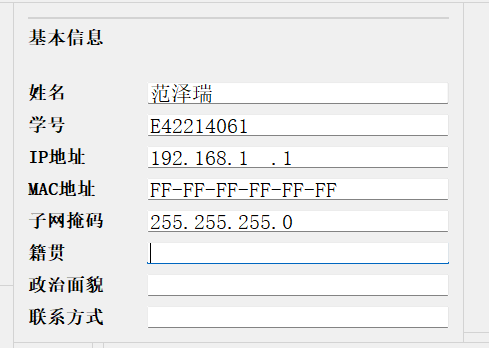
控件功能展示：

1. 选择年纪专业：





1. 基本信息输入：



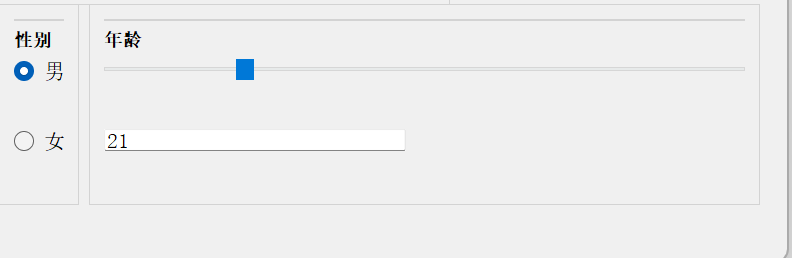
IP地址的输入采用格式校验，要求必须输入0-9的数字，可忽略。

并且使用正则表达式对其校验，保证符合IP格式

MAC地址采用格式校验，智能输入16进制字符，不可忽略。

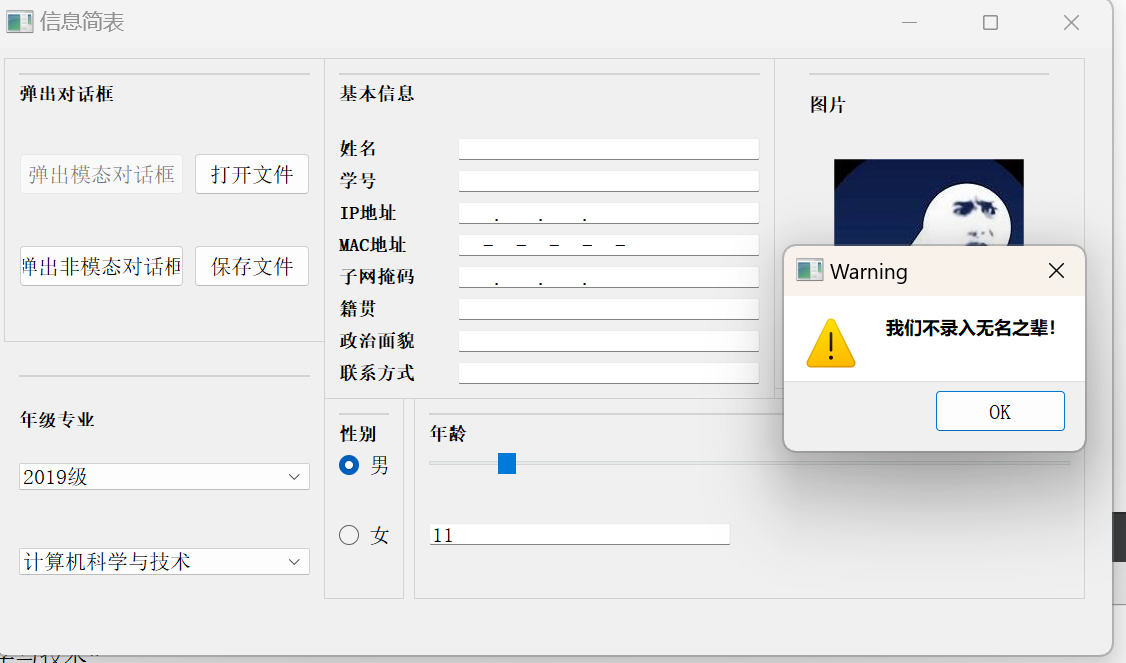
子网掩码校验同IP地址

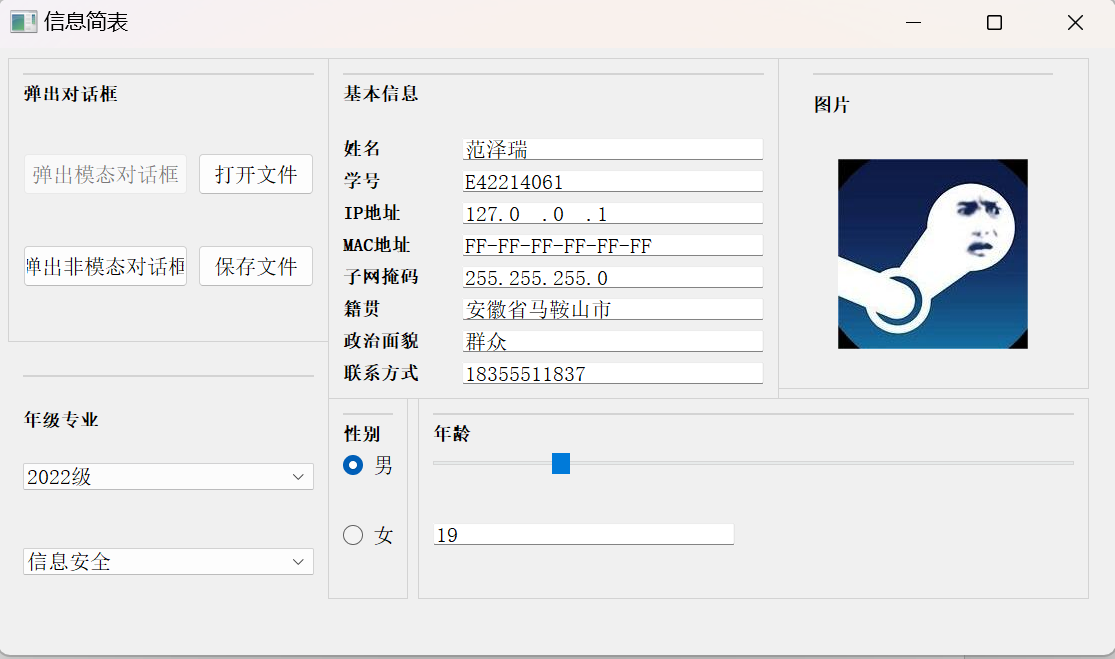
1. 性别与年龄

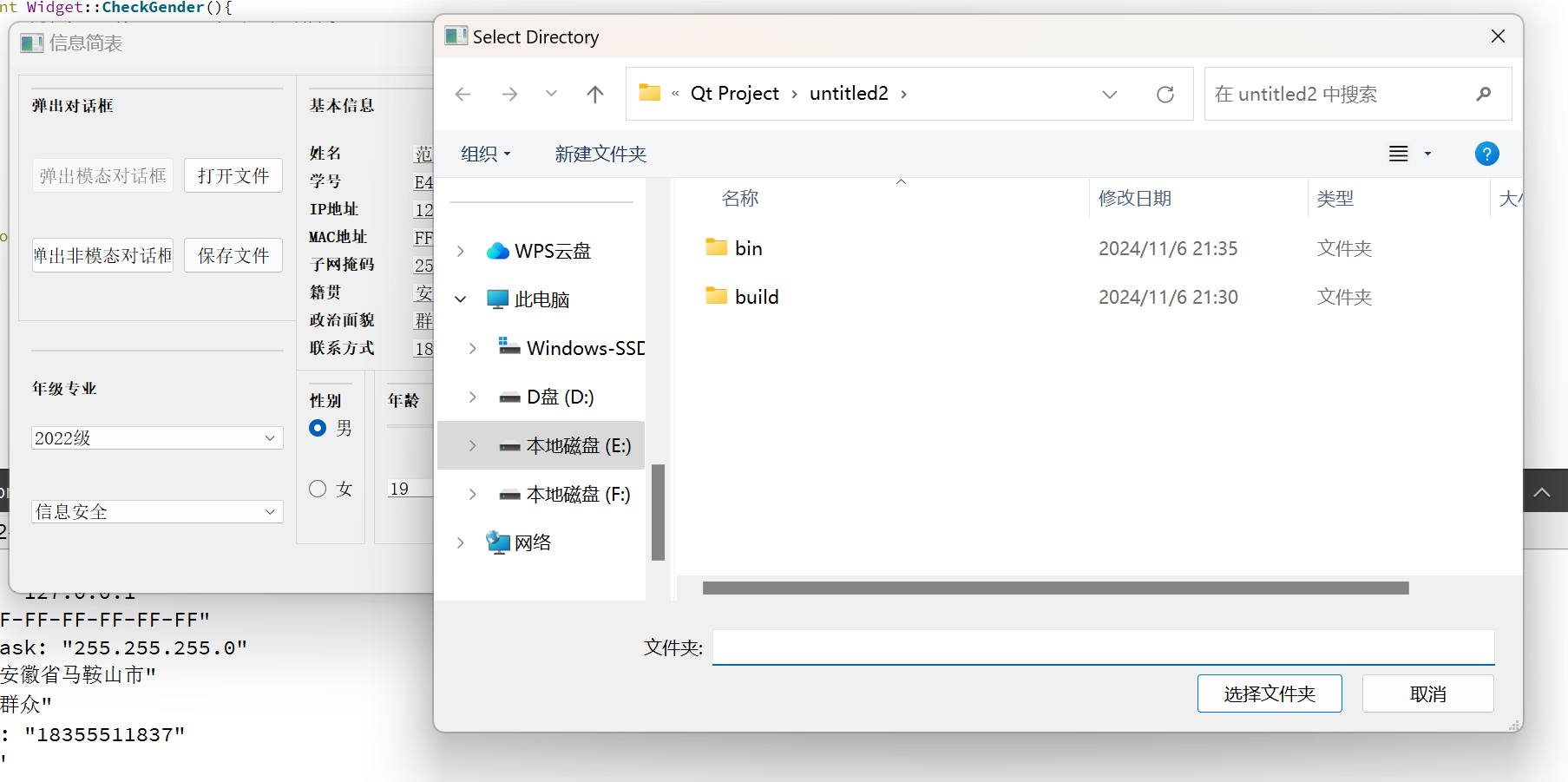


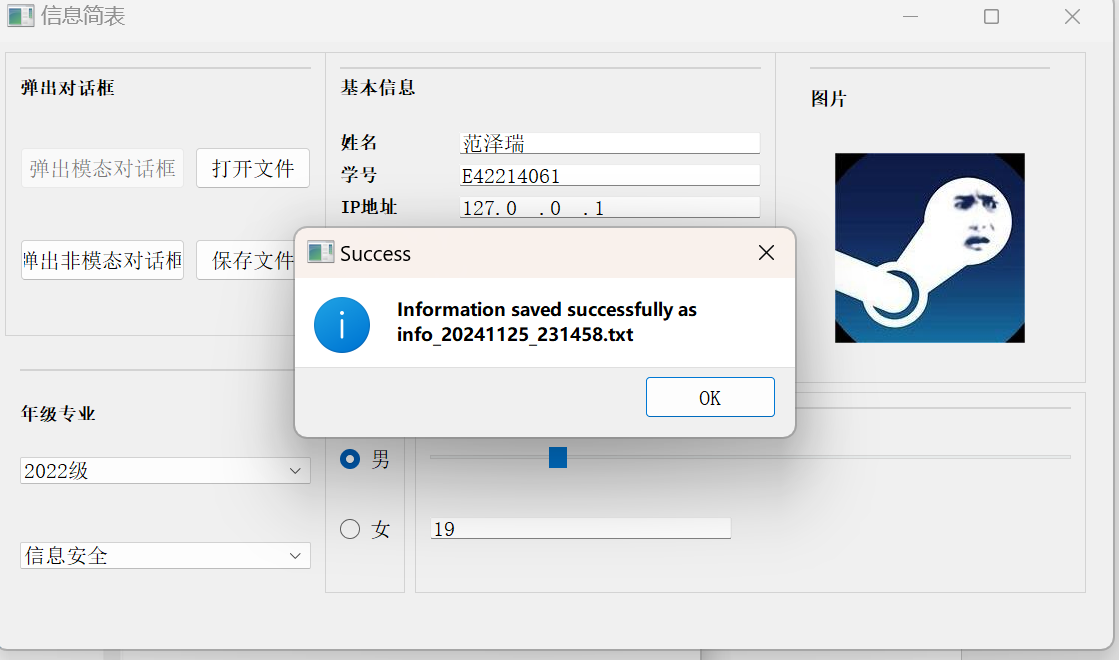
通过radiobox控件选择男女；通过slider控件设置年龄

1. 保存输入信息为文件：

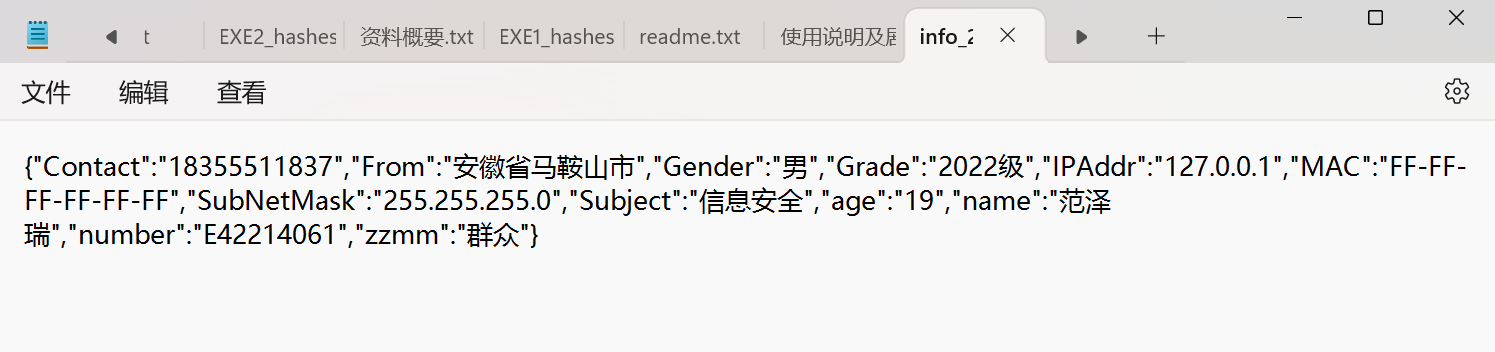




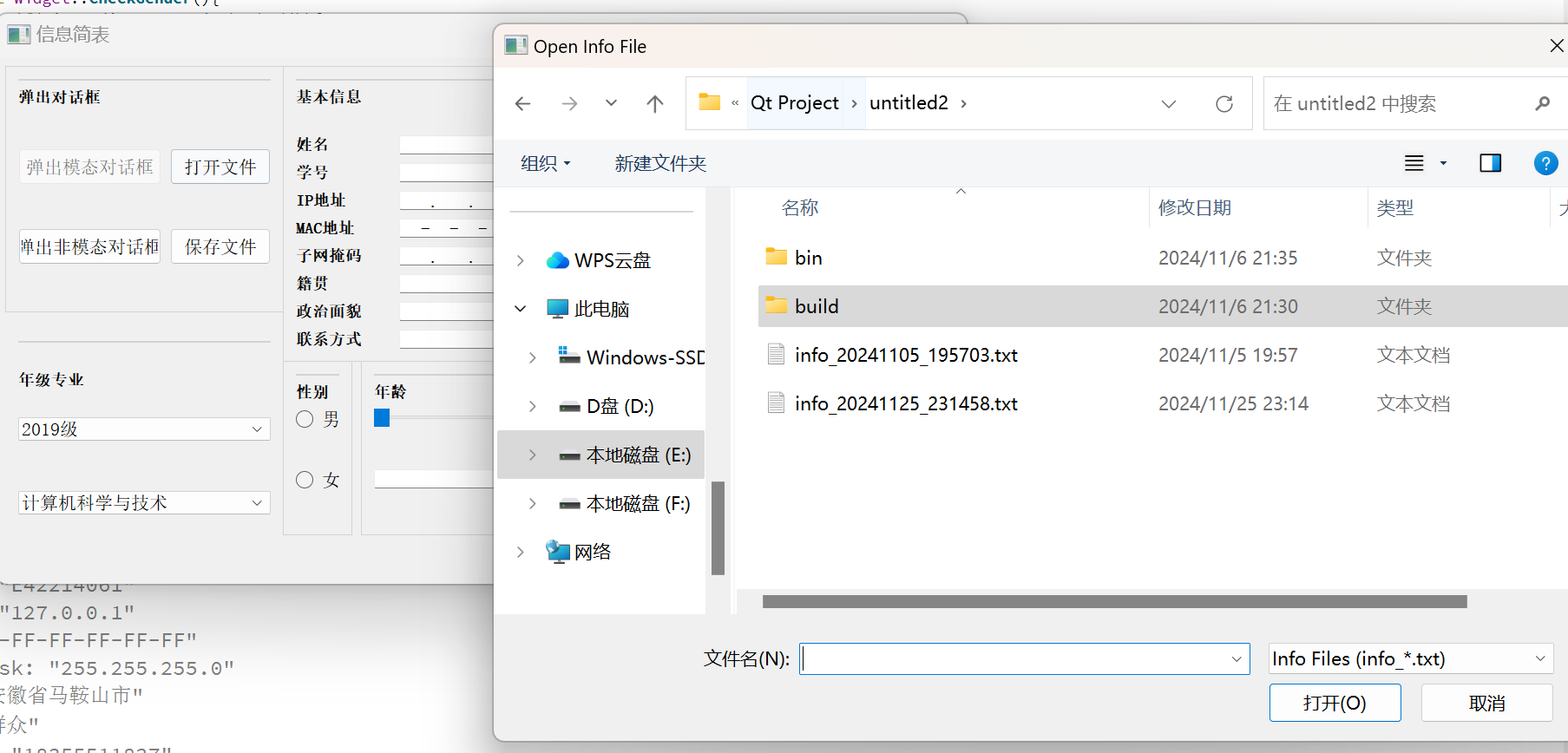


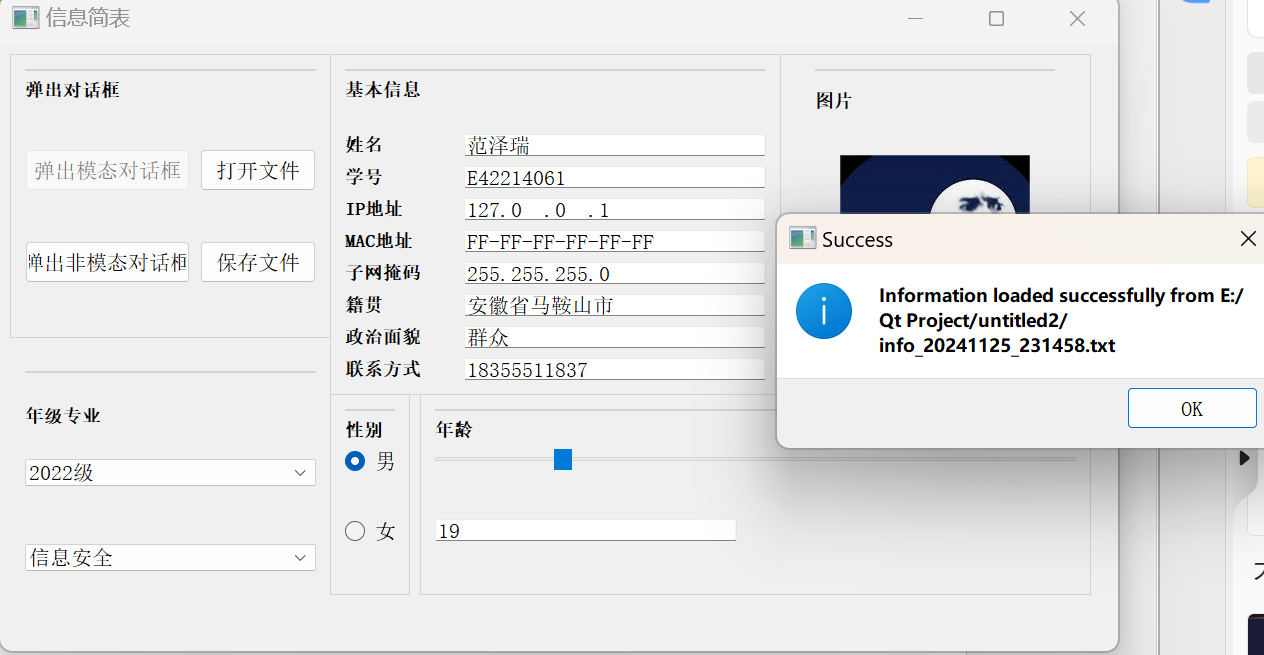


文件内容：



1. 打开读取文件：





保存文件代码：

void Widget::StoreInfo(){

QJsonObject jsonObj;//创建json对象存储个人信息

//从各个控件中获取个人信息

QString Grade = ui->comboBox->currentText();

Grade.toLocal8Bit().constData();//!!!!!对中文转码

jsonObj["Grade"]=Grade;

qDebug()<<Grade;

QString Subject = ui->comboBox\_2->currentText();

Subject.toLocal8Bit().constData();

jsonObj["Subject"]=Subject;

qDebug()<<Subject;

QString Gender;

switch(CheckGender()){

case 0:

Gender = QString::fromLocal8Bit("男");

break;

case 1:

Gender = QString::fromLocal8Bit("女");

break;

case 2:

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("我们不支持LGBT！")); // 警告信息

return;

break;

}

jsonObj["Gender"]=Gender;

qDebug()<<Gender;

if(ui->lineEdit\_10->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("AGE！"));

return;

}// 警告信息

jsonObj["age"]=ui->lineEdit\_10->text();

qDebug()<<ui->lineEdit\_10->text();

if(ui->lineEdit->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("我们不录入无名之辈！"));

return;

}

jsonObj["name"]=ui->lineEdit->text();//!!!从控件中输入的中文可以不用转码

if(ui->lineEdit\_2->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("number"));

return;

}

jsonObj["number"]=ui->lineEdit\_2->text();

if(ui->lineEdit\_3->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("IP！"));

return;

}

QString IPAddr = ui->lineEdit\_3->text().remove(QRegExp("^ +\\s\*")); //使用正则表达式去除空格

jsonObj["IPAddr"]=IPAddr;//IP地址

if(ui->lineEdit\_6->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("MAC！"));

return;

}

jsonObj["MAC"]=ui->lineEdit\_6->text();//物理地址

if(ui->lineEdit\_4->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("subnet"));

return;

}

jsonObj["SubNetMask"]=ui->lineEdit\_4->text().remove(QRegExp("^ +\\s\*"));//子网掩码

if(ui->lineEdit\_5->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("籍贯！"));

return;

}

jsonObj["From"]=ui->lineEdit\_5->text();//籍贯

if(ui->lineEdit\_8->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("政治面貌?"));

return;

}

jsonObj["zzmm"]=ui->lineEdit\_8->text();

if(ui->lineEdit\_9->text().isEmpty()){

QMessageBox::warning(this, "Warning", QString::fromLocal8Bit("大隐隐于市？"));

return;

}

jsonObj["Contact"]=ui->lineEdit\_9->text();

QJsonDocument jsonDoc(jsonObj);

QByteArray byteArray = jsonDoc.toJson(QJsonDocument::Compact); //转为字节数组，方便写入到文件

QString selectedDir = QFileDialog::getExistingDirectory(this, tr("Select Directory"), "",

QFileDialog::ShowDirsOnly | QFileDialog::DontResolveSymlinks);//打开选择路径对话框

selectedDir.toLocal8Bit().constData(); //!!!!!十分重要，对中文路径进行转码】

if (selectedDir.isEmpty()) {

QMessageBox::warning(this, tr("Warning"), tr("No directory selected."));

return;

}

// 获取当前时间并格式化为字符串,唯一标识文件名

QString timestamp = QDateTime::currentDateTime().toString("yyyyMMdd\_HHmmss");

QString fileName = QString("info\_%1.txt").arg(timestamp);//拼接，组成文件名

// 构建完整的文件路径

QString filePath = QDir(selectedDir).filePath(fileName);

qDebug() << "File Path:" << filePath;

QFile file(filePath);

if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {

QMessageBox::warning(this, tr("Warning"), tr("Cannot open file for writing: ") + file.errorString());

return;

}

// 写入 QByteArray 到文件

qint64 bytesWritten = file.write(byteArray);

if (bytesWritten == -1) {

QMessageBox::warning(this, tr("Warning"), tr("Failed to write data to file."));

file.close();

return;

}

file.close();

qDebug() << "Data written to" << filePath;

QMessageBox::information(this, tr("Success"), tr("Information saved successfully as ") + fileName);

return;

}

void Widget::on\_pushButton\_2\_clicked()//打开文件

{

QString fileName = QFileDialog::getOpenFileName(this, tr("Open Info File"), "",

tr("Info Files (info\_\*.txt)")); //打开选择文件对话框

if (fileName.isEmpty()) {

QMessageBox::warning(this, tr("Warning"), tr("No file selected."));

return;

}

//打开并读取文件内容

QFile file(fileName);

if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {

QMessageBox::warning(this, tr("Warning"), tr("Cannot open file for reading: ") + file.errorString());

return;

}

QByteArray byteArray = file.readAll();

file.close();

QJsonParseError jsonError;

QJsonDocument jsonDoc = QJsonDocument::fromJson(byteArray, &jsonError);

if (jsonError.error != QJsonParseError::NoError) {

QMessageBox::warning(this, tr("Warning"), tr("Failed to parse JSON data: ") + jsonError.errorString());

return;

}

if (!jsonDoc.isObject()) {

QMessageBox::warning(this, tr("Warning"), tr("Invalid JSON format: not an object."));

return;

}

QJsonObject jsonObj = jsonDoc.object();//创建json对象，接收从文件中读取的信息

QString Grade = jsonObj.value("Grade").toString();

QString Subject = jsonObj.value("Subject").toString();

QString Gender = jsonObj.value("Gender").toString();

QString age = jsonObj.value("age").toString();

QString name = jsonObj.value("name").toString();

QString number = jsonObj.value("number").toString();

QString IPAddr = jsonObj.value("IPAddr").toString();

QString MAC = jsonObj.value("MAC").toString();

QString SubNetMask = jsonObj.value("SubNetMask").toString();

QString From = jsonObj.value("From").toString();

QString zzmm = jsonObj.value("zzmm").toString();

QString Contact = jsonObj.value("Contact").toString();

// qDebug() << "Grade:" << Grade;

// qDebug() << "Subject:" << Subject;

// qDebug() << "Gender:" << Gender;

// qDebug() << "Age:" << age;

// qDebug() << "Name:" << name;

// qDebug() << "Number:" << number;

// qDebug() << "IPAddr:" << IPAddr;

// qDebug() << "MAC:" << MAC;

// qDebug() << "SubNetMask:" << SubNetMask;

// qDebug() << "From:" << From;

// qDebug() << "zzmm:" << zzmm;

// qDebug() << "Contact:" << Contact;

//设置各个控件的信息

ui->comboBox->setCurrentText(Grade);

ui->comboBox\_2->setCurrentText(Subject);

if (Gender == QString::fromLocal8Bit("男")) {

ui->radioButton->setChecked(true);

} else if (Gender == QString::fromLocal8Bit("女")) {

ui->radioButton\_2->setChecked(true);

}

ui->lineEdit\_10->setText(age);

ui->lineEdit->setText(name);

ui->lineEdit\_2->setText(number);

ui->lineEdit\_3->setText(IPAddr);

ui->lineEdit\_6->setText(MAC);

ui->lineEdit\_4->setText(SubNetMask);

ui->lineEdit\_5->setText(From);

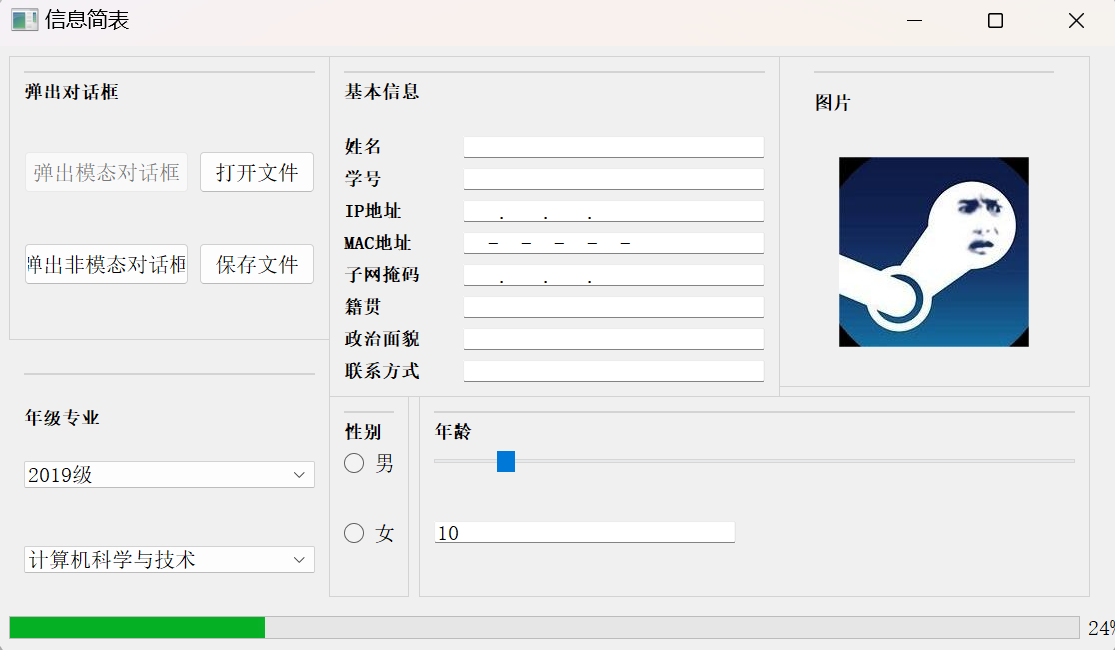
ui->lineEdit\_8->setText(zzmm);

ui->lineEdit\_9->setText(Contact);

QMessageBox::information(this, tr("Success"), tr("Information loaded successfully from ") + fileName);

}

进度条：



采用QProgressBar控件实现。

**2.基本的数据运算和处理—计算器**

2.1 功能要求

简易计算(必做)：实现两数的加、减、乘、除、、取模等简单计算；

科学计算：实现包括(反)正弦、(反)余弦、(反)正切、(反)余切、乘方、开方、指数等函数运

算；

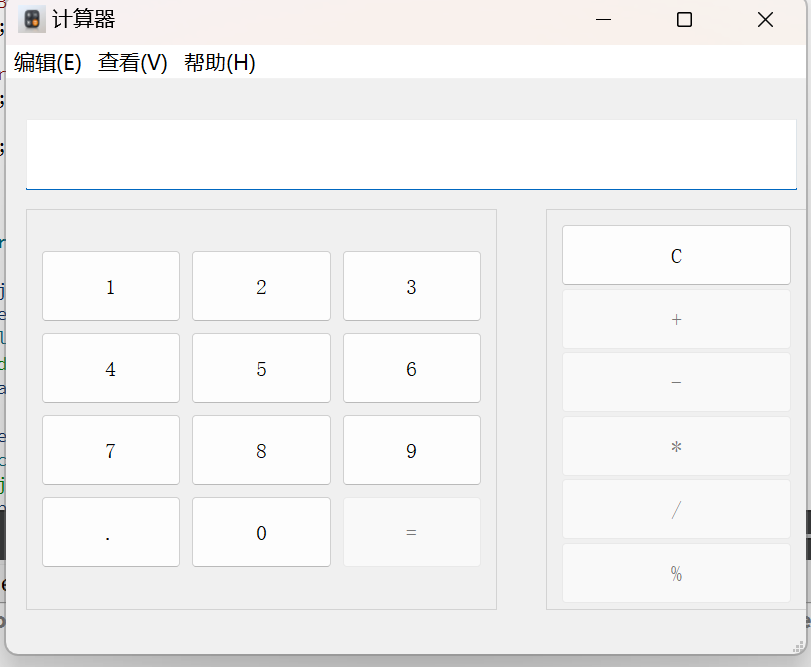
提示功能：具备数据的溢出错误提示，无意义运算表达式提示等；

进制转换：实现二进制、八进制、十进制、十六进制转换功能。系统的用户界面如图 2(简单计算器)和图 3(科学计算器)所示。所有运算符号和数字使用按键

输入，实时显示。

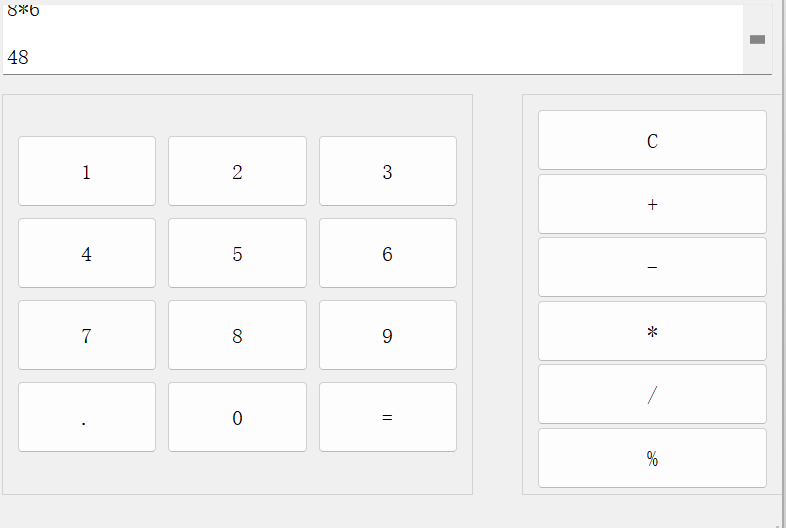
简易计算器实现：

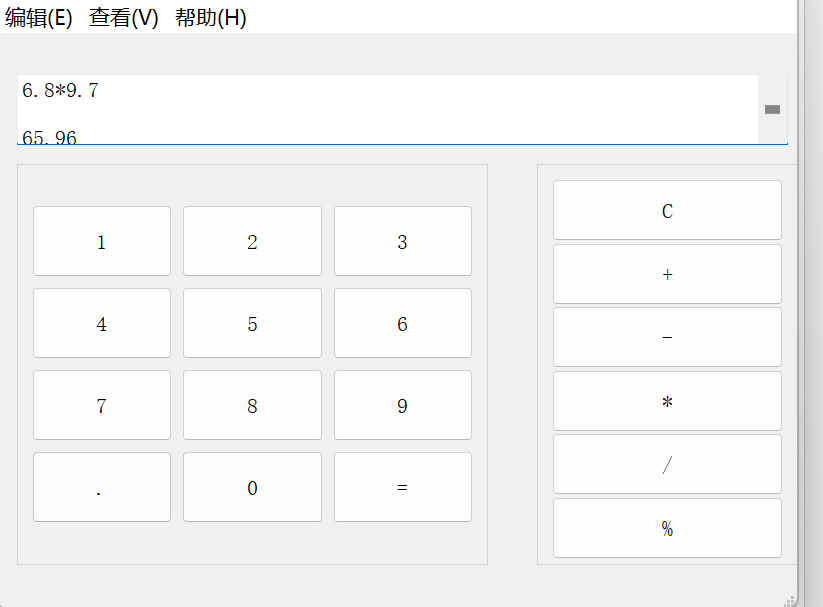
UI界面：

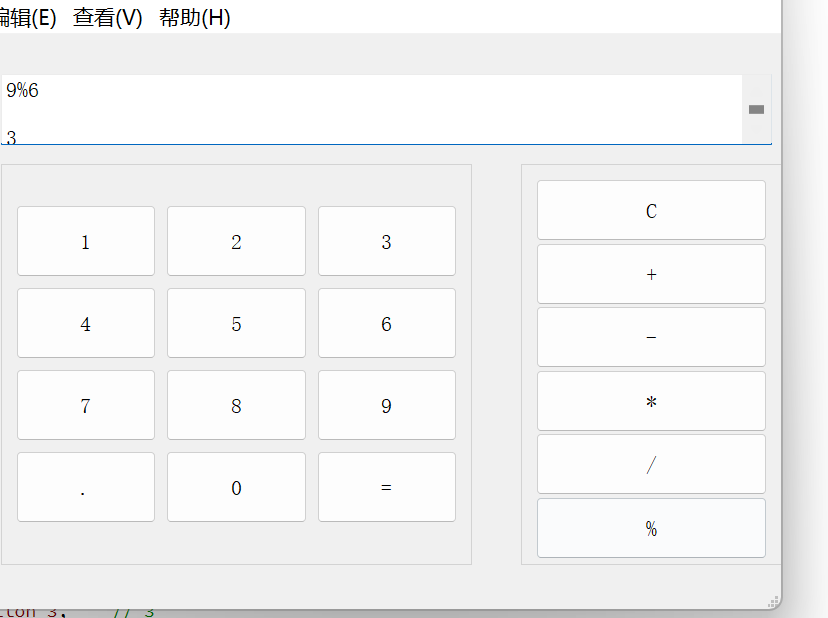


功能演示：

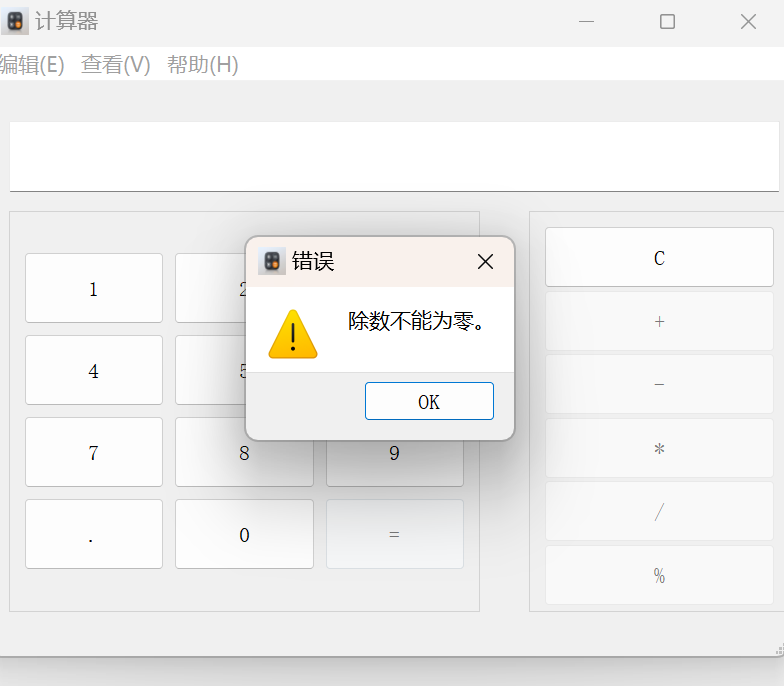
当用户没有输入时，禁用运算功能按钮







当进行模运算与除法运算时，若检测到除数为0，弹窗警告



具体求值过程：

使用栈实现逆波兰式表达式求值

代码：

double MainWindow::evaluateExpression(const QString &expr, bool &ok) // 表达式求值

{

// 使用栈实现逆波兰表达式求值

QStack<double> operands;

QStack<QChar> operators;

//使用lambda表达式

// 定义运算符优先级

auto precedence = [](QChar op) -> int { //返回类型为int

if(op == '+' || op == '-') return 1;

if(op == '\*' || op == '/' || op == '%') return 2;

return 0;

};

// 定义运算函数

auto applyOp = [&](QChar op, double a, double b) -> double {

switch(op.toLatin1()){

case '+': return a + b;

case '-': return a - b;

case '\*': return a \* b;

case '/':

if(b != 0){

return a / b;

} else {

ok = false; // 设置错误标志

return 0; // 返回默认值

}

case '%':

if(b != 0){

return static\_cast<int>(a) % static\_cast<int>(b);

} else {

ok = false; // 设置错误标志

return 0; // 返回默认值

}

default:

ok = false; // 未知运算符，设置错误标志

return 0;

}

};

// 初始化 ok 为 true

ok = true;

// 解析表达式

QRegExp tokenRegex("(\\d+\\.\\d+)|(\\d+)|([+\\-\*/%])");//解析数字和运算符

int pos = 0;

while(pos < expr.length()){

if(expr[pos].isSpace()){ //遇到空格跳过

pos++;

continue;

}

if(expr[pos].isDigit() || expr[pos] == '.'){

// 解析数字

if(tokenRegex.indexIn(expr, pos) == pos){ //成功匹配一个数字

QString numberStr = tokenRegex.cap(0);//捕获

bool okNum;

double number = numberStr.toDouble(&okNum);

if(!okNum){

ok = false;

return 0;

}

operands.push(number);//入栈

pos += numberStr.length();//移动下标

}

else{

ok = false;

return 0;

}

}

else if(QString("+-\*/%").contains(expr[pos])){

//解析运算符

QChar currentOp = expr[pos];

while(!operators.isEmpty() && precedence(operators.top()) >= precedence(currentOp)){//如果栈顶运算符的优先级大于或等于当前运算符的优先级（>=），则需要先对栈中的运算符进行运算（因为优先级高的运算符需要先执行）

if(operands.size() < 2){//是否有足够的操作数（至少需要2个）

ok = false;

return 0;

}

//出栈

double b = operands.pop();

double a = operands.pop();

QChar op = operators.pop();

double res = applyOp(op, a, b);

//新的运算数入栈

operands.push(res);

if(!ok){

return 0;

}

}

operators.push(currentOp);

pos++;//移动下标，处理下一个字符

}

else{

// 无效字符

ok = false;

return 0;

}

}

// 处理剩余的运算符

while(!operators.isEmpty()){

if(operands.size() < 2){

ok = false;

return 0;

}

double b = operands.pop();

double a = operands.pop();

QChar op = operators.pop();

double res = applyOp(op, a, b);

operands.push(res);

if(!ok){

return 0;

}

}

if(operands.size() != 1){ //最后运算数栈中应该只剩下一个元素

ok = false;

return 0;

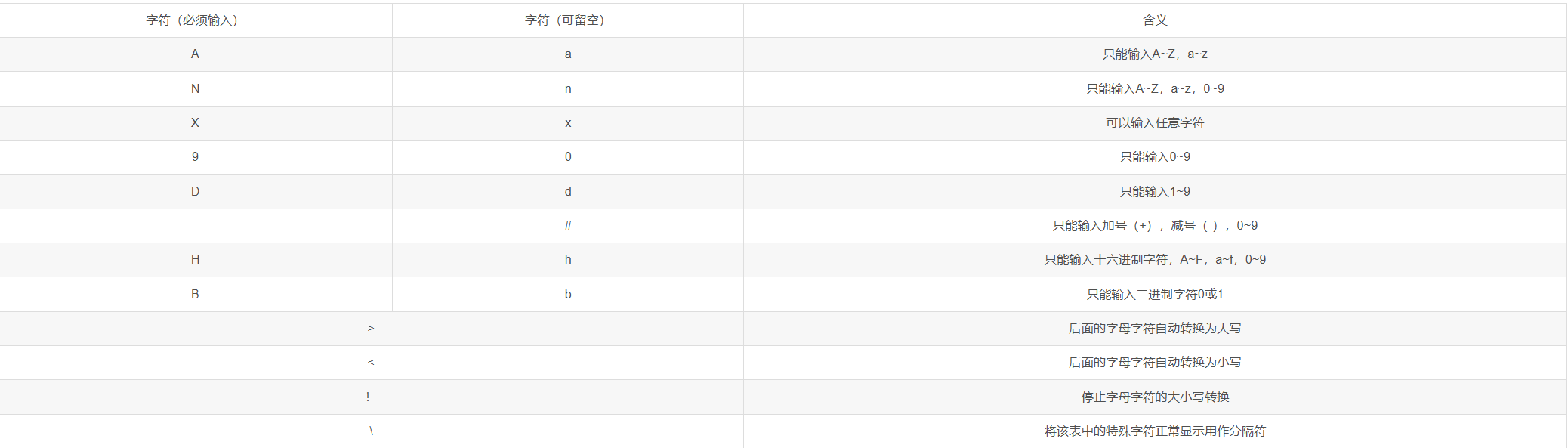
}

return operands.pop();

}

**【实验总结】**

1. 在制作对话框进行IP，子网掩码，MAC地址输入时，为保证输入正确的格式，可以采用setInputMask设置输入掩码。



例如通过setInputMask("000.000.000.000;");//0表示可忽略，且输入只能是0-9 设置IP与子网掩码的输入，确保只能输入数字，同时每三位间用.分隔。

1. IP地址的格式校验可采用正则表达式，QString ipPattern = QString("^%1\\.%2\\.%3\\.%4$").arg(octet).arg(octet).arg(octet).arg(octet);
2. 打开选择文件对话框采用QFileDialog::getOpenFileName打开文件对话框，选择文件路径。该方法第一个参数为父对象，通常设为this；第二个参数为对话框标题；第三个参数为默认路径；第四个参数为过滤器，限制用于限制用户可见的文件类型，本实例中采用tr("Info Files (info\_\*.txt)")作为限制。获取文件路径后，使用QFile的open方法打开文件，使用readall方法读取文件内容（返回类型QByteArray）。创建QJsonDocument对象，使用QJsonDocument::fromJson方法，将字节数组转为Json文件。再通过QJsonDocument的object方法，转为QJsonObj对象，使用value方法通过键提取相应的值，填充到ui控件中。
3. 保存文件：创建QJsonObj对象，采用json格式封装个人信息。信息封装完成后，将QJsonObj对象封装为QJsonDocment对象。再通过QJsonDocument的的toJson方法（参数采用QJsonDocument::Compact作为模式）生成QByteArray字节数组，方便写入到文件。通过QFileDialog::getExistingDirectory方法选择路径。使用时间戳生成文件名后与选择的路径进行拼接生成完整文件路径。通过QFile的write方法写入到文件中。
4. 实现计算器时，核心算法在于对输入的表达式进行求值。这里参考逆波兰式求值的方式。创建运算数栈与运算符栈。通过扫描表达式，将数字入栈，将运算符按优先级处理。运算符根据优先级处理：如果栈顶运算符优先级较高或相等，出栈进行计算；否则将当前运算符压入 operators 栈。运算时，从运算数栈取出两数，运算后再放入栈中。循环处理，直到运算数栈中只剩下一个元素，即为表达式结果。