**程序设计开发实践报告**

**题 目 中国象棋**

**序 号 01**

**学 号 20232241091**

**姓 名 蔡佳兴**

**任课教师 范晓娅，于鲲鹏**

**成 绩**

大连理工大学软件学院

2024年7月

**目录**

[评分细则及标准 2](#_Toc143947979)

[1 技术调研报告 3](#_Toc143947980)

[1.1 第一周学习总结](#_Toc143947981) 3

[1.1.1 内容简介](#_Toc143947982) 3

[1.1.2 难点和解决办法 1](#_Toc143947983)3

[1.1.3 学习案例 1](#_Toc143947984)3

**评分细则及标准**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **考察项目** | **总分** | **评分细则** | **分数** | **得分** |
| **技术调研（第一周）** | 20分 | 调研报告 | 10分 |  |
| 平时表现 | 10分 |
|  |  |  |  |  |
| **技术调研（第二周）** | 20分 | 调研报告 | 10分 |  |
| 平时表现 | 10分 |
|  | | | | |
| **项目开发评分（总分60分）** | | | |  |
| 问题规模 | 10分 | 创新超额完成指定任务，工作量饱满 | 8-10分 |  |
| 基本完成指定任务，工作量一般 | 6-8分 |
| 指定任务未完成，工作量不足 | 0-5分 |
| 技术难度 | 10分 | 模型设计合理优化，采用合适的方法实现 | 8-10分 |  |
| 模型设计基本正确，采用较合适的方法实现 | 6-8分 |
| 模型设计存在问题，采用方法不合理 | 0-5分 |
| 实现程度 | 20分 | 系统实现完整，界面友好，测试全面无误 | 8-10分 |  |
| 系统实现完整，界面友好，存在少许错误 | 6-8分 |
| 系统实现不完整，界面不友好，存在错误 | 0-5分 |
| 报告质量 | 10分 | 报告完整、格式统一、结构清晰、图表正确 | 8-10分 |  |
| 报告较为规范、结构较为清晰、图表基本正确 | 6-8分 |
| 内容不完整不规范、结构不清晰、图表有错误 | 0-5分 |
| 项目汇报 | 10分 | 阐述清晰准确，回答问题准确到位 | 8-10分 |  |
| 阐述基本准确，回答问题基本准确 | 6-8分 |
| 阐述不够清晰完整，回答问题不准确 | 0-5分 |
|  |  |  |  |  |
| 最终得分 | 100分 |  | | |

# 1 技术调研报告

## 1.1 第一周学习总结

### 1.1.1 内容简介

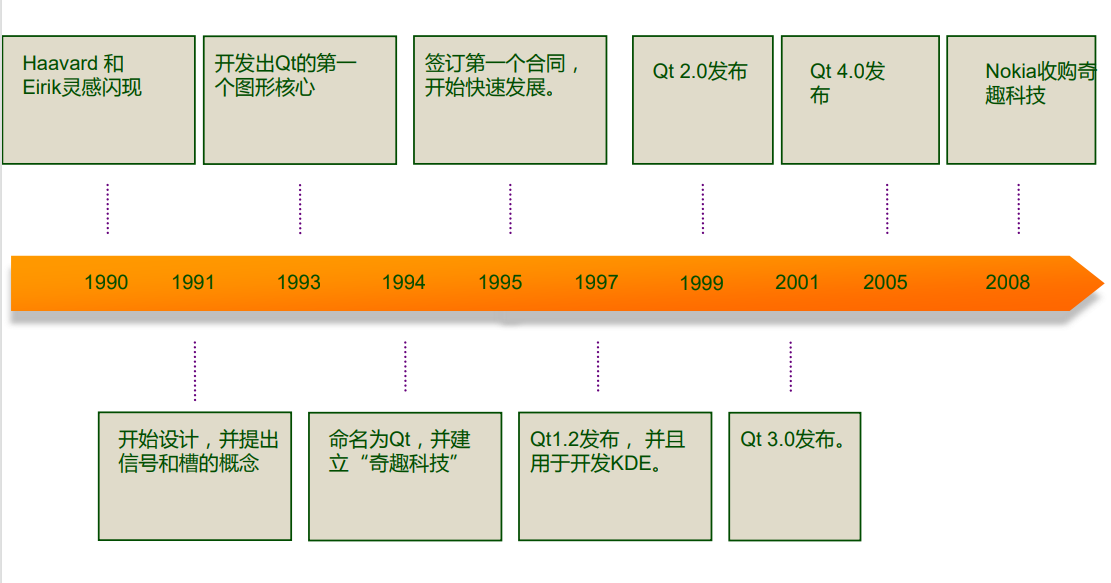
• 由挪威Trolltech公司开发，目前已被Nokia收购。

• Qt是一个基于C++的跨平台应用程序和UI开发框架。

• 它包含一个类库，和用于跨平台开发及国际化的工具。

• 开源KDE桌面的基石。

发展史：



窗口部件：

顶级窗口：一个没有父窗口的窗口部件被叫做顶级窗口部件。

//新建QWidget类对象

//默认parent参数是0

//所以它是个窗

QWidget \*widget = new QWidget(0, Qt::Dialog);

子部件：指定了父窗口的部件为子部件。

//label2指定了父窗口为widget

//所以不是窗口

QLabel \*label2 = new QLabel(widget);

非模态对话框：

方法：

QDialog \*dialog = new QDialog(this);

dialog->setModal(false);

dialog->show();

模态对话框：

方法一：

Q Dialog \*dialog = new QDialog(this);

dialog->setModal(true);

di alog->show();

方法二：

QDialog dialog(this);

dialog.setModal(true); dialog.exec();



对话框常用函数：

exec函数

– 调用exec()来显示模式对话框。当用户关闭这个对话框，exec() 将提供一个可用的返回值并且这时流程控制继续从调用exec() 的地方进行。

accept()槽

– 在使用模式对话框,隐藏模式对话框并且设置结果代码为 Accepted。 

reject ()槽

– 隐藏模式对话框并且设置结果代码为Rejected。 

show()

– 调用show()来显示非模式对话框，show()立即返回。

颜色对话框：

// 创建对象

QColorDialog dialog(Qt::red,this);

// 显示alpha选项

dialog.setOption(QColorDialog::ShowAlphaChan nel);

// 以模态方式运行对话框

dialog.exec();

// 获取当前颜色

QColor color = dialog.currentColor();

文件对话框：

// 文件对话框

QStringList fileNames = QFileDialog::getOpenFileNames( this,tr(“文件对话框”), //标题

“F:”, //初始路径

tr(“图片文件(\*png \*jpg)”)); //文件类型

字体对话框：

// 标记是否按下了“OK”按钮 bool ok;

// 获得选择的字体

QFont font = QFontDialog::getFont(&ok,this);

// 如果按下“OK”按钮

//那么让“字体对话框”按钮使用新字体

// 如果按下“Cancel”按钮

//那么输出信息

if(ok)

ui->pushButton\_3->setFont(font);

else

qDebug() <<tr("没有选择字体！");

输入对话框：

// 获取字符串

QString string = QInputDialog::getText(this,tr("输入字符串对话框"), tr("请输入用户名："),QLineEdit::Normal,tr("admin"),&ok);

// 获取整数

int value1 = QInputDialog::getInt(this,tr("输入整数对话框"), tr("请输入-1000到1000之间的数值 "),100,-1000,1000,10,&ok);

// 获取浮点数

double value2 = QInputDialog::getDouble(this,tr("输入浮点数 对话框"), tr("请输入-1000到1000之间的数值 "),0.00,-1000,1000,2,&ok);

// 获取条目

QStringList items; items << tr("条目1") << tr("条 目2"); QString item = QInputDialog::getItem(this,tr("输入条目对话 框"), tr("请选择一个条目"),items,0,true,&ok);

消息对话框：

// 问题对话框

int ret1 = QMessageBox::question(this,tr("问题对话框"), tr("你了 解Qt吗？"),QMessageBox::Yes,QMessageBox::No); if(ret1 == QMessageBox::Yes) qDebug()<<tr("问题！");

// 提示对话框

int ret2 = QMessageBox::information(this,tr("提示对话框"), tr(" 这是Qt书籍！"),QMessageBox::Ok); if(ret2 == QMessageBox::Ok) qDebug()<<tr("提示！");

// 警告对话框

int ret3 = QMessageBox::warning(this,tr("警告对话 框"), tr("不能提前结束！"),QMessageBox::Abort); if(ret3 == QMessageBox::Abort) qDebug()<<tr("警告！");

// 错误对话框

int ret4 = QMessageBox::critical(this,tr("严重错误对话框"), tr(“发现一个严重错误！现在要关闭所有文件！ "),QMessageBox::YesAll); if(ret4 == QMessageBox::YesAll) qDebug() <<tr("错误");

// 关于对话框

QMessageBox::about(this,tr("关于对话框"), tr("yafeilinux.com 致力于Qt及Qt Creator的普及工作！"));

进度对话框：

// 进度对话框

QProgressDialog dialog(tr("文件复制进度"),tr("取消 "),0,50000,this); // 设置窗口标题

dialog.setWindowTitle(tr("进度对话框"));

// 将对话框设置为模态

dialog.setWindowModality(Qt::WindowModal);

dialog.show();

// 演示复制进度

for(int i=0;i<50000;i++)

{

// 设置进度条的当前值 dialog.setValue(i);

// 避免界面冻结 QCoreApplication::processEvents();

// 按下取消按钮则中断 if(dialog.wasCanceled()) break;

}

// 这样才能显示100%，因为for循环中少加了一个数dialog.setValue(50000);

错误信息对话框：

//错误信息对话框

QErrorMessage \*dialog = new QErrorMessage(this);

dialog->setWindowTitle(tr("错误信息对话框"));

dialog->showMessage(tr("这里是出错信息！"));

向导对话框：

// 向导对话框

void MyWidget::on\_pushButton\_8\_clicked() {

QWizard wizard(this);

wizard.setWindowTitle(tr("向导对话框"));

// 添加向导页面

wizard.addPage(createPage1());

wizard.addPage(createPage2());

wizard.addPage(createPage3());

wizard.exec();

}

按钮部件：QAbstractButton是按钮部件的抽象基类。

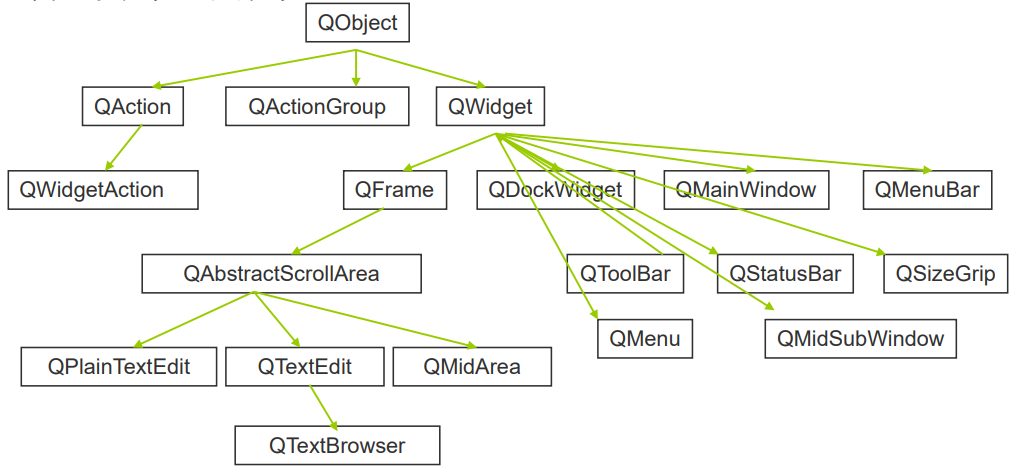
型编辑器：行编辑器 QLineEdit是单行文本编辑器。

数值设定框：QAbstractSpinBox是数值设定框的基类，提供数值设定方法。

滑块部件：QAbstractSlider提供一个滑块，定位到某范围内整数任意值。

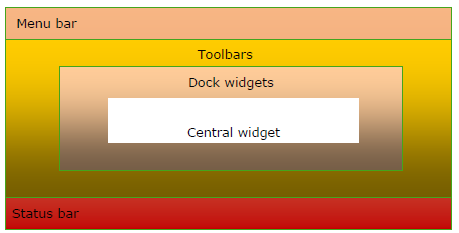
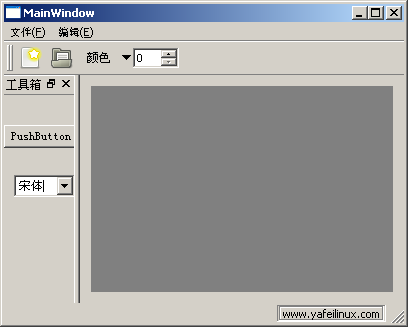
**应用程序主窗口**

窗口类的继承体系



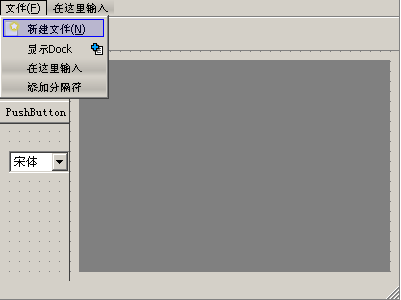
### **主程序窗口框架**

一个widget就是一个窗口， Qt提供一个QMainWindow类 管理主窗口程序



菜单栏：

Desinger左上角双击“从这里输入”，开始编辑菜单项在ActionEditor配置菜单项。



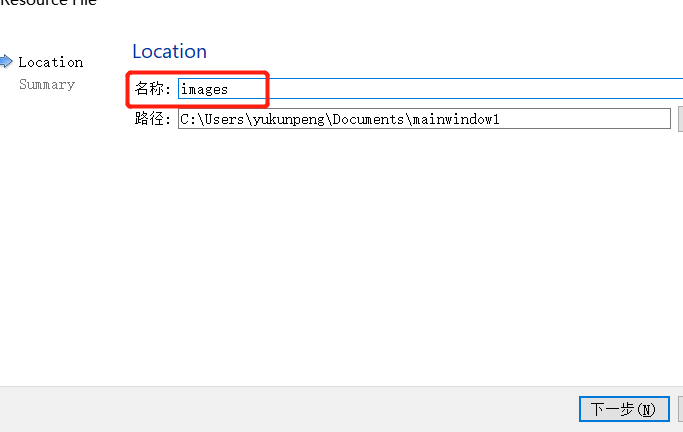
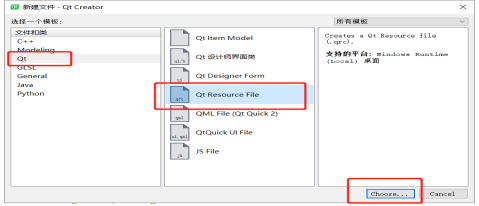
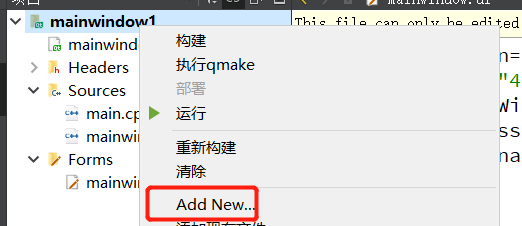
### 项目资源：

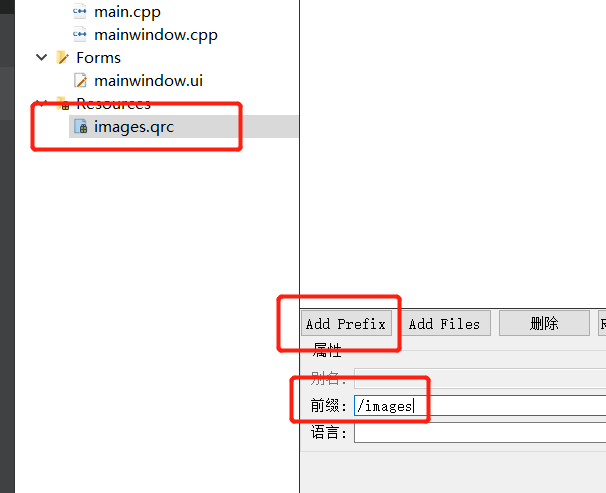
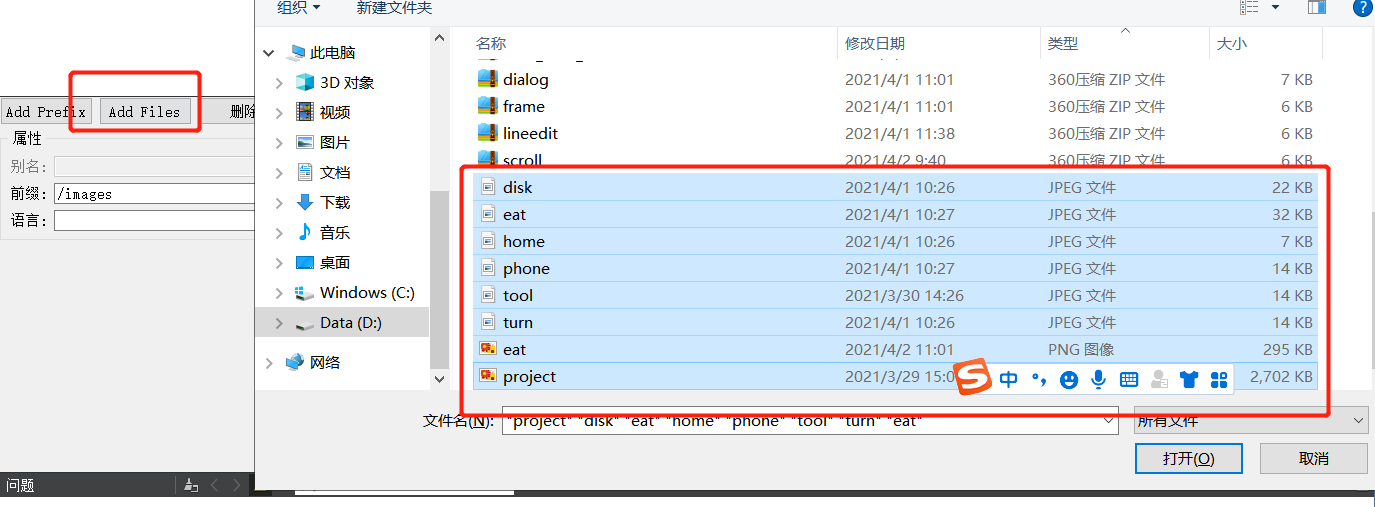
1.工程上点右键->addNew选择Qt->QtResouceFiles

2.给资源起名

3.项目中选择images.qrc->AddPrefix->修改前缀名（前缀名，相当于资源目录名）

4.选择AddFiles，选取工程的资源文件



设置菜单项的槽函数：

Private slots:

void on\_actionOpen\_triggered();

通过代码编辑菜单：

// 添加编辑菜单

QMenu \*editMenu = ui->menuBar->addMenu(tr("编辑(&E)"));

// 添加打开菜单

QAction\*action\_OpeneditMenu->addAction( QIcon(":/myImage/images/open.png"),tr("打开文件(&O)"));

// 设置快捷键

action\_Open->setShortcut(QKeySequence("Ctrl+O"));

QToolBar：包含了一组控件的可移动面板

// 在工具栏中添加动作

ui->mainToolBar->addAction(action\_Open);

// 向工具栏添加QToolButton按钮

ui->mainToolBar->addWidget(toolBtn);

// 创建QSpinBox

QSpinBox \*spinBox = new QSpinBox(this);

// 向工具栏添加QSpinBox部件

ui->mainToolBar->addWidget(spinBox);

中心部件:

在窗口的中心区域放置一个中心部件,对于文档程序来讲一般是一个QTextEdit

// 新建文本编辑器部件

QTextEdit \*edit = new QTextEdit(this);

// 使用QMdiArea类的addSubWindow()函数

//创建子窗口，以文本编辑器为中心部件

QMdiSubWindow \*child = ui->mdiArea->addSubWindow(edit);

child->setWindowTitle(tr("多文档编辑器子窗口"));

// 显示子窗口

child->show();

Dock部件：

QDockWidget 可以停靠在中心部件的旁边，可以悬浮在主窗口上面，就像一个工具箱，放置其他部件。

状态栏：

状态信息分为三类：临时信息，一般信息，永久信息。

临时信息： showMessage

一般信息： addWidget

永久信息： addPermanentWidget

// 显示临时消息，显示2000毫秒即2秒钟

ui->statusBar-> showMessage(tr("欢迎"),2000);

// 创建标签

QLabel \*permanent = new QLabel(this);

// 标签样式

permanent->setFrameStyle( QFrame::Box | QFrame::Sunken);

// 显示信息 permanent->setText(“welcom");

// 将标签设置为永久部件

ui->statusBar-> addPermanentWidget(permanent);

自定义菜单：

创建继承QWidgetAction的类

class MyAction : public QWidgetAction

实现createWidget虚函数，添加部件

QWidget\* createWidget(QWidget \*parent)

富文本处理：

富文本—文档中可以使用颜色，字体，表格等多种格式。

QTextEdit是一个高级的所见即所得的支持使用HTML风格标记 的多信息文本格式的查看器/编辑器。

文本块：

一个文本块可以看作一个段落，回车换行可以看作一个新的文本块。

QTextDocument \*document = ui->textEdit->document();

// 获取文档的第一个文本块

QTextBlock block = document->firstBlock();

for(int i=0;

iblockCount(); i++){ qDebug() <<

tr("文本块%1， 文本块首行行号为:%2， 长度为:%3,内容为：")

.arg(i)

.arg(block.firstLineNumber())

.arg(block.length())

<< block.text();

// 获取下一个文本块

block = block.next();

表格 列表 图片：

编辑器中插入表格

QTextCursor cursor(ui->textEdit->textCursor()); cursor.movePosition(QTextCursor::Start);

QTextTable \*table = cursor.insertTable(3,2);

编辑器中插入表格列表

QTextCursor cursor = ui->textEdit->textCursor(); cursor.movePosition(QTextCursor::NoMove); cursor.insertList(QTextListFormat::ListDecimal);

编辑器中插入图片

QTextCursor cursor = ui->textEdit->textCursor(); cursor.movePosition(QTextCursor::NoMove); QTextImageFormat format; format.setName("d://phone.jpeg"); format.setWidth(100); format.setHeight(100); cursor.insertImage(format);

拖放打开文件：

事件：

拖动进入事件（drag）

void MainWindow::dragEnterEvent

放下事件 （drop）

void MainWindow::dropEvent

事件函数：

接收动作：acceptProposedAction

忽略该事件：ignore

dragEnterEvent代码：

if (event->mimeData()->hasFormat("text/uri-list"))

{ event->acceptProposedAction(); }

dropEvent代码：

if(event->mimeData()->hasUrls())

{ QList list = event->mimeData()->urls();

foreach(QUrl url, list)

{ QFile file(url.toLocalFile());

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

QByteArray t = file.readAll(); ui->textEdit->append(QString(t)); file.close(); } }

自定义拖放：

事件：

鼠标按下事件

MainWindow::mousePressEvent

拖动进入事件

void MainWindow::dragEnterEvent

拖动事件

void MainWindow::dragMoveEvent

放下事件

void MainWindow::dropEvent

打印文档：

打印机对象

QPrinter

打印对话框

QPrintDialog

选中区域

QPrintDialog::addEnabledOption

按下了打印按钮

QPrintDialog ::Accepte

QPrinter printer;

QPrintDialog dialog;

// 如果文本编辑器中有选择区域则打印选择区域

if(ui->textEdit->textCursor().hasSelection())

{ dialog.addEnabledOption(QAbstractPrintDialog::PrintSelection); }

if(dialog.exec() == QDialog::Accepted)

{ ui->textEdit->print(&printer); }

打印预览：

创建打印预览对话框

QPrintPreviewDialog

需要关联相关的信号槽

connect(&dialog ,

&QPrintPreviewDialog::paintRequested ,

this ,

&MainWindow::printPreview);

dialog.exec()

打印PDF：

在Pro文件中追加 printsupport

QPrinter printer;

// 指定输出格式为pdf

printer.setOutputFormat(QPrinter::PdfFormat); printer.setOutputFileName(fileName);

ui->textEdit->print(&printer);

HTTP协议：

### 1.1.2 难点和解决办法

一、

问题：创建Qt Widget Appication，运行后报错：

 09:14:35: Could not determine which "make" command to run. Check the "make" step in the build configuration.

 Error while building/deploying project test1 (kit: Desktop Qt 5.10.0 MSVC2013 64bit)

 When executing step "qmake"

原因：我的电脑中没有VS2013编译器，所以无法编译；

解决办法：在qt主界面中选择项目--Buld&Run--Desktop Qt 5.10.0 MSVC2015 32bit（或者Desktop Qt 5.10.0 MSVC2015 64bit，因为我的编译器是VS2015，选择电脑中存在的编译即可）；

待跟进：使用Desktop Qt 5.10.0 MSVC2015 32bit和Desktop Qt 5.10.0 MSVC2015 64bit来编译会有什么差别？

二、

问题：在程序运行时报错：代码中有不能编辑的字符；

原因：Qtcreator中有些中文不能编辑，导致编译出错；

解决办法：用英文代替中文（还没找到使中文能很好地显示的办法）；

待跟进：使中文能再Qtcreator中能顺利地显示出来；

### 1.1.3 学习案例

# **实现QQ界面：**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~***MainWindow***()

{

delete ui;

}

class Drawer : public QToolBox

initToolButton

#ifndef DRAWER\_H

#define DRAWER\_H

#include <QWidget>

#include <QToolBox>//工具盒类，层叠窗口，抽屉效果实现

#include <QToolButton>

class Drawer : public QToolBox

{

Q\_OBJECT

public:

Drawer(QWidget \*parent=0,Qt::WindowFlags f = 0);

~Drawer();

void initToolButton(QToolButton \*\*tb,QString name,QString pathpic);

private :

QToolButton \*toolbtn1;

QToolButton \*toolbtn2;

QToolButton \*toolbtn3;

QToolButton \*toolbtn4;

QToolButton \*toolbtn5;

QToolButton \*toolbtn11;

QToolButton \*toolbtn12;

QToolButton \*toolbtn21;

QToolButton \*toolbtn22;

};

#endif // DRAWER\_H

#include "drawer.h"

#include<QGroupBox>

#include <QVBoxLayout>

#include <QDebug>

Drawer::Drawer(QWidget \*parent, Qt::WindowFlags f):QToolBox(parent,f)

{

setWindowTitle("QQ");

initToolButton(&toolbtn1,"张远山","1.png");

qDebug()<<"ok";

initToolButton(&toolbtn2,"东奥","2.png");

qDebug()<<"ok";

initToolButton(&toolbtn3,"孙选成","3.png");

qDebug()<<"ok";

initToolButton(&toolbtn4,"司宽","4.png");

qDebug()<<"ok";

initToolButton(&toolbtn5,"张宇","5.png");

qDebug()<<"ok";

QGroupBox \*gb1 = new QGroupBox;

QVBoxLayout \*lay1 = new QVBoxLayout(gb1);

lay1->setMargin(10);

lay1->setAlignment(Qt::AlignHCenter);

lay1->addWidget(toolbtn1);

lay1->addWidget(toolbtn2);

lay1->addWidget(toolbtn3);

lay1->addWidget(toolbtn4);

lay1->addWidget(toolbtn5);

// lay1->addStretch();

initToolButton(&toolbtn11,"刘翔","11.png");

initToolButton(&toolbtn12,"詹姆斯","12.png");

QGroupBox \*gb2 = new QGroupBox;

QVBoxLayout \*lay2 = new QVBoxLayout(gb2);

lay2->addWidget(toolbtn11);

lay2->addWidget(toolbtn12);

lay2->setMargin(10);

lay2->setAlignment(Qt::AlignHCenter);

initToolButton(&toolbtn21,"惠子","21.png");

initToolButton(&toolbtn22,"名人","22.png");

QGroupBox \*gb3 = new QGroupBox;

QVBoxLayout \*lay3 = new QVBoxLayout(gb3);

lay3->addWidget(toolbtn21);

lay3->addWidget(toolbtn22);

lay3->setMargin(10);

lay3->setAlignment(Qt::AlignHCenter);

//准备好的抽屉插入TOOlBox中

this->addItem(gb1,"我的好友");

this->addItem(gb2,"陌生人");

this->addItem(gb3,"黑名单");

}

Drawer::~Drawer(){}

void Drawer::initToolButton(QToolButton \*\*tb, QString name, QString pathpic)

{

(\*tb) = new QToolButton;

(\*tb)->setText(name);

(\*tb)->setIcon(QPixmap(pathpic));

(\*tb)->setIconSize(QSize(20,20));

(\*tb)->setAutoRaise(true);

(\*tb)->setToolButtonStyle(Qt::ToolButtonTextBesideIcon);

}

运行结果：