## 附

南京工程学院

**实 训 报 告**

课程名称 **多媒体编程基础**

实训项目名称 实训 2： 文本编辑器设计

实训学生班级 数嵌 172

实训学生姓名 朱广锋

学　　　　号 202170638

|  |  |  |
| --- | --- | --- |
| 1. **实训目的**   1.掌握 Qt Creator 的基本使用方法  2.理解界面设计的相关类  3.设计界面并编写一个简单的文本编辑程序，功能尽量完整  **二、实训环境及开发工具：**  PC机、Qt5.14（或其它版本）  **三、实训要求及内容：**  1. 练习各种界面风格的设计方式。  2. 编写一个简单的文本编辑程序，功能尽量完整。  3. 可以参考教材或提供的源代码，自己选择开发方法。  **四、程序设计思路（30分）**  本次程序设计，我采用了程序员更习惯使用的markdown作为编辑器的文档格式。  为了方便编辑与预览，界面分左右栏来分别进行编辑代码和预览格式。编辑页面为QPlainTextEdit，预览界面为QTextBrowser，使用Qt默认的markdown渲染方式，并在编辑界面更新时同步预览界面。  在菜单栏下方设置了工具栏，可以快速的设置文字格式等操作，对文本格式的编辑都会转换为markdown语法格式。  字体字号颜色等格式的修改可以使用快捷按钮或者菜单的对话框，也支持直接书写markdown语法。  程序的界面（主窗口、查找窗口、关于窗口）主要在Qt Designer中进行，部分简单的信号逻辑也在其中完成。  程序初始化会预先写入简单的markdown文档内容，以便于预览格式。  **五、设计方法及代码（30分）**  程序主要的信号逻辑部分在NotepadXWindow的构造函数中完成，使用lambda表达式作为接收槽。   |  | | --- | | // 内容同步  connect(ui.markdownEdit, &QPlainTextEdit::textChanged, [this] { ui.markdownPreview->setMarkdown(ui.markdownEdit->toPlainText()); });  // 查找  connect(ui.actionFind, &QAction::triggered, [this] {  if (ui.markdownEdit->textCursor().hasSelection())  {  findDialog.ui.lineEdit->setText(ui.markdownEdit->textCursor().selectedText());  }  findDialog.show();  });  // 查找下一个  connect(findDialog.ui.pushButton, &QPushButton::clicked, [this] {  QString findText = findDialog.ui.lineEdit->text();  if (ui.markdownEdit->find(findText))  {  QPalette palette = ui.markdownEdit->palette();  palette.setColor(QPalette::Highlight, palette.color(QPalette::Active, QPalette::Highlight));  ui.markdownEdit->setPalette(palette);  }  else  {  QMessageBox::information(this, "NotepadX", "没有查找到内容", QMessageBox::Ok);  }  });  // 加粗  connect(ui.actionBold, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  ui.markdownEdit->textCursor().insertText("\*\*" + selectedText + "\*\*");  }  });  // 斜体  connect(ui.actionItalic, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  ui.markdownEdit->textCursor().insertText("\*" + selectedText + "\*");  }  });  // 下划线  connect(ui.actionUnderline, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  ui.markdownEdit->textCursor().insertText("<u>" + selectedText + "</u>");  }  });  // 删除线  connect(ui.actionStrikeout, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  ui.markdownEdit->textCursor().insertText("~~" + selectedText + "~~");  }  });  // 颜色  connect(ui.actionColor, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  auto color = QColorDialog::getColor(Qt::black, this);  if (color.isValid())  {  ;  selectedText = "<font color=" + color.name() + ">" + selectedText + "</font>";  ui.markdownEdit->textCursor().insertText(selectedText);  }  }  });  // 字体  connect(ui.actionFont, &QAction::triggered, [this]  {  if (ui.markdownEdit->textCursor().hasSelection())  {  auto selectedText = ui.markdownEdit->textCursor().selectedText();  bool ok;  auto font = QFontDialog::getFont(&ok, this);  if (ok)  {  selectedText = "<font face=\"" + font.family() + "\" size=" + QString::number(font.pointSize()) + ">" + selectedText + "</font>";  ui.markdownEdit->textCursor().insertText(selectedText);  }  }  });  // 打开  connect(ui.actionOpen, &QAction::triggered, [this]  {  auto fileName = QFileDialog::getOpenFileName(this, "打开文件", "/", "Markdown文档(\*.md)");  QFile file(fileName);  if (file.exists())  {  file.open(QIODevice::ReadOnly | QIODevice::Text);  if (file.isReadable())  {  workingFileName = fileName;  ui.markdownEdit->setPlainText(QString(file.readAll()));  }  file.close();  }  });  // 保存  connect(ui.actionSave, &QAction::triggered, [this]  {  if (workingFileName.isEmpty())  workingFileName = QFileDialog::getSaveFileName(this, "保存文件", "/", "Markdown文档(\*.md)");  QFile file(workingFileName);  file.open(QIODevice::WriteOnly | QIODevice::Text);  if (file.isWritable())  file.write(ui.markdownEdit->toPlainText().toUtf8());  file.close();  });  // 保存为  connect(ui.actionSaveAs, &QAction::triggered, [this]  {  workingFileName = QFileDialog::getSaveFileName(this, "保存为", "/", "Markdown文档(\*.md)");  QFile file(workingFileName);  file.open(QIODevice::WriteOnly | QIODevice::Text);  if (file.isWritable())  file.write(ui.markdownEdit->toPlainText().toUtf8());  file.close();  });  // 关于  connect(ui.actionAbout, &QAction::triggered, &about, &About::exec);  // 打印 TODO: 未完全完成  connect(ui.actionPrint, &QAction::triggered, [this]  {  QPrinter printer;  QPrintDialog\* dialog = new QPrintDialog(&printer, this);  dialog->setWindowTitle("打印");  if (ui.markdownPreview->textCursor().hasSelection())  dialog->addEnabledOption(QAbstractPrintDialog::PrintSelection);  if (dialog->exec() != QDialog::Accepted)  return;  }); |   初始化内容   |  | | --- | | // 填充初始内容  ui.markdownEdit->setPlainText(R"MARKDOWN(# 标题1  ## 标题2  ### 标题3  #### 标题4  ##### 标题5  ###### 标题6  ---  # 2. 列表  \* 圆圈  - 实心圆  + 方块  ---  # 3. 嵌套列表  1. 第一项：  - 第一项嵌套的第一个元素  - 第一项嵌套的第二个元素  1. 第二项：  - 第二项嵌套的第一个元素  - 第二项嵌套的第二个元素  ---  # 4. 任务列表  - [ ] task1  - [x] task2  - [x] task3  ---  # 5. 文字格式  \*\*粗体文本\*\*  \_斜体文本\_  \_\*\*粗斜体文本\*\*\_  ~~删除线文本~~  <u>带下划线文本</u>  <font face="微软雅黑" >微软雅黑字体</font>  <font size=1 >1号字</font>  <font size=2 >2号字</font>  <font color=#FF0000 >红色</font>  <font color=#008000 >绿色</font>  <font color=#0000FF >蓝色</font>  `code`  ```cpp  #include <QApplication>  #include "NotepadXWindow.h"  int main(int argc, char\* argv[])  {  QApplication a(argc, argv);  NotepadXWindow mainWindow;  mainWindow.show();  return a.exec();  }  ```  ---  # 6. 分隔线  ---  # 7. 表格  | 表头1 | 表头2 |这里是右对齐|这里是左对齐|  | ---- | :----: |---:|:---|  | 单元格1 | 单元格2 |3|-3|  | 单元格4 | 单元格5 |6|-6|  )MARKDOWN"); |   **六、实训结果及说明（30分）**  1. 主界面内容如下，左为编辑窗口，右为预览窗口。  2. 菜单栏，包括文件【新建、打开、保存、另存为、打印】，编辑【复制、剪切、粘贴、撤销、重做】，格式【加粗、斜体、下划线、删除线、颜色、字体】，查看【放大、缩小】。  3. 查找功能(Ctrl+F)，会自动填充当前选择内容或使用上次查找内容。  4. 打印  5. 颜色，修改的颜色会以markdown语法格式插入（见下图）  6. 字体，修改的字体字号会以markdown语法插入（见下图）  7. 文件操作，下图为 保存为 的示例。保存：保存到当前打开的文件（如没有，则弹出选择保存位置），保存为：保存到新的文件，并设定为当前打开。  8. 关于，可点击文字引导至主页。  **七、实训思考（10分）**  问题：  1. Qt对与markdown的渲染不够完善，对于图片格式![]()无法显示其内容，可以尝试使用QWebEngineView+js来进行渲染；   1. 打印、恢复默认缩放等部分功能没有完全实现； 2. 未保存文件在退出时没有保存提示； 3. 按钮图标的粗细有区别，风格不是完全统一。   拓展：   1. 编辑和预览没有实现同步滚动，应该增加一个开关来设定是否同步滚动； 2. 可以将当前编辑的文件显示在窗口标题上。 |