**《数据结构综合设计》**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 名称 | | 作业2 Qt&&数据结构 | | 评分1 |  | 评分2 |  | 总分 |  |
| 类型 | | □验证型 ☑设计型 □综合型 | |
| 专业班级 | 卓软2401 | 学号 | 5120246728 |
| 任课教师 | 杨春明 | 姓名 | 华昊朗 | 时间 | | | 2025-07-01 | | |
| 说明 | 评分1针对教学大纲课程目标2，满分10分；评分2针对课程目标3，满分5分。本次作业满分15分。 | | | | | | | | |

# 一、目的

1、通过实现窗口布局、控件交互和信号槽机制，深入理解 Qt 界面开发的核心流程，熟悉 Qt Designer 可视化工具与代码实现的协同工作模式，掌握简单的GUI程序设计基础知识。

2、通过数据结构与Qt 编程有机结合，进一步提高应用特定数据结构完成程序设计的能力。

1. 能够根据程序设计的需求，具体练使用主流开发工具的能力和不断学习新技术的能力。

# 原理

对于这个项目，我们综合任务二和任务三，在通过使用QTextEdit作为核心文本编辑组件实现记事本功能的基础上，使用垂直布局作为主布局管理器，嵌套水平布局和网格布局组织不同区域的小部件，实现元素尺寸随窗口大小自动变化。同时，我们支持将文件保存为多种文件格式，覆盖大部分主流编程语言，通用的txt以及Markdown格式。回退操作基本使用的QTextEdit自带，不出意外使用的是栈结构。

# 三、环境

1、操作系统：Windows 11

2、集成开发环境：Qt 6.9，Qt Creator 17.0.0

3、编程语言：C++

# 四、内容与步骤

# 因篇幅限制，暂省略ui设计部分代码。

## 1、定义头文件mainwindow.h

|  |
| --- |
| #ifndef MAINWINDOW\_H  #define MAINWINDOW\_H  #include <QMainWindow>  #include <QCloseEvent>  #include <QAction>  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\_action\_triggered(); // 新建  void on\_action\_2\_triggered(); // 打开  void on\_action\_3\_triggered(); // 保存  void on\_action\_4\_triggered(); // 另存为  void on\_action\_5\_triggered(); // 退出  void on\_action\_6\_triggered(); // 回退  void on\_action\_7\_triggered(); // 取消回退  // 右键菜单动作  void cutText();  void copyText();  void pasteText();  void selectAllText();  protected:  void closeEvent(QCloseEvent \*event) override;  private:  Ui::MainWindow \*ui;  QString currentFilePath;  bool isModified = false;  // 右键菜单动作  QAction \*cutAction;  QAction \*copyAction;  QAction \*pasteAction;  QAction \*selectAllAction;  void setupContextMenu();  void loadFile(const QString &filePath);  void saveFile(const QString &filePath);  void updateWindowTitle();  QString getFileFilter();  };  #endif // MAINWINDOW\_H |

## 2、主函数入口main.cpp

|  |
| --- |
| #include "mainwindow.h"  #include <QApplication>  int main(int argc, char \*argv[])  {  QApplication a(argc, argv);  MainWindow w;  w.show();  return a.exec();  } |

## 3、功能函数mainwindow.cpp

|  |
| --- |
| #include "mainwindow.h"  #include "ui\_mainwindow.h"  #include <QFileDialog>  #include <QFile>  #include <QTextStream>  #include <QMessageBox>  #include <QCloseEvent>  #include <QFileInfo>  #include <QSize>  #include <QMenu>  MainWindow::MainWindow(QWidget \*parent)  : QMainWindow(parent)  , ui(new Ui::MainWindow)  {  ui->setupUi(this);  // 最小尺寸600\*400  setMinimumSize(600, 400);  updateWindowTitle();  connect(ui->textEdit, &QTextEdit::textChanged, [this] {  isModified = true;  updateWindowTitle();  });  isModified = false;  currentFilePath = "";  setupContextMenu();  }  MainWindow::~MainWindow()  {  delete ui;  }  // 右键菜单  void MainWindow::setupContextMenu()  {  ui->textEdit->setContextMenuPolicy(Qt::CustomContextMenu);  connect(ui->textEdit, &QTextEdit::customContextMenuRequested, [this](const QPoint &pos) {  QMenu \*menu = ui->textEdit->createStandardContextMenu();  menu->exec(ui->textEdit->mapToGlobal(pos));  delete menu;  });  }  void MainWindow::cutText()// 剪切  {  ui->textEdit->cut();  }  void MainWindow::copyText()// 复制  {  ui->textEdit->copy();  }  void MainWindow::pasteText()// 粘贴  {  ui->textEdit->paste();  }  void MainWindow::selectAllText()// 全选  {  ui->textEdit->selectAll();  }  QString MainWindow::getFileFilter()  {  return "文本文件 (\*.txt);;"  "Markdown 文件 (\*.md);;"  "HTML 文件 (\*.html \*.htm);;"  "CSS 文件 (\*.css);;"  "JavaScript 文件 (\*.js);;"  "C 文件 (\*.c);;"  "C++ 文件 (\*.cpp \*.cxx \*.cc);;"  "Python 文件 (\*.py);;"  "Go 文件 (\*.go);;"  "C# 文件 (\*.cs);;"  "Swift 文件 (\*.swift);;"  "所有文件 (\*);;";  }  void MainWindow::updateWindowTitle()  {  QString title = "小白记事本";  if (!currentFilePath.isEmpty()) {  title += " - " + QFileInfo(currentFilePath).fileName();  }  if (isModified) {  title += " \*";  }  setWindowTitle(title);  }  void MainWindow::loadFile(const QString &filePath)  {  QFile file(filePath);  if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {  QMessageBox::warning(this, "打开文件", "无法打开文件: " + file.errorString());  return;  }  QTextStream in(&file);  ui->textEdit->setText(in.readAll());  file.close();  currentFilePath = filePath;  isModified = false;  updateWindowTitle();  }  void MainWindow::saveFile(const QString &filePath)  {  QFile file(filePath);  if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {  QMessageBox::warning(this, "保存文件", "无法保存文件: " + file.errorString());  return;  }  QTextStream out(&file);  out << ui->textEdit->toPlainText();  file.close();  currentFilePath = filePath;  isModified = false;  updateWindowTitle();  }  void MainWindow::on\_action\_triggered()  {  if (isModified) {  QMessageBox::StandardButton reply;  reply = QMessageBox::question(this, "保存更改", "当前文档已修改，是否保存更改？",  QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);  if (reply == QMessageBox::Save) {  on\_action\_3\_triggered();  if (isModified) return;  } else if (reply == QMessageBox::Cancel) {  return;  }  }  ui->textEdit->clear();  currentFilePath = "";  isModified = false;  updateWindowTitle();  }  void MainWindow::on\_action\_2\_triggered()  {  if (isModified) {  QMessageBox::StandardButton reply;  reply = QMessageBox::question(this, "保存更改", "当前文档已修改，是否保存更改？",  QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);  if (reply == QMessageBox::Save) {  on\_action\_3\_triggered()  if (isModified) return;  } else if (reply == QMessageBox::Cancel) {  return;  }  }  QString filePath = QFileDialog::getOpenFileName(  this, "打开文件", "", getFileFilter()  );  if (!filePath.isEmpty()) {  loadFile(filePath);  }  }  void MainWindow::on\_action\_3\_triggered()  {  if (currentFilePath.isEmpty()) {  on\_action\_4\_triggered();  return;  }  saveFile(currentFilePath);  }  void MainWindow::on\_action\_4\_triggered()  {  QString filePath = QFileDialog::getSaveFileName(  this, "另存为", "", getFileFilter()  );  if (!filePath.isEmpty()) {  QFileInfo fileInfo(filePath);  if (fileInfo.suffix().isEmpty()) {  filePath += ".txt";//默认扩展名txt  }  saveFile(filePath);  }  }  // 退出  void MainWindow::on\_action\_5\_triggered()  {  close();  }  // 回退  void MainWindow::on\_action\_6\_triggered()  {  ui->textEdit->undo();  }  // 取消回退  void MainWindow::on\_action\_7\_triggered()  {  ui->textEdit->redo();  }  // 关闭  void MainWindow::closeEvent(QCloseEvent \*event)  {  if (isModified) {  QMessageBox::StandardButton reply;  reply = QMessageBox::question(this, "保存更改", "当前文档已修改，是否保存更改？",QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);  if (reply == QMessageBox::Save) {  on\_action\_3\_triggered();  if (isModified) {  event->ignore();  } else {  event->accept();  }  } else if (reply == QMessageBox::Discard) {  event->accept();  } else {  event->ignore();  }  } else {  event->accept();  }  } |

# 五、结果

# 经调试，基本实现了所给要求，用户可以在主页面进行正常输入，可以使用右键进行功能性操作，并可以通过工具栏实现基本功能，以及保存为主流常用的格式。

# 电脑萤幕的截图 AI 生成的内容可能不正确。电脑萤幕的截图 AI 生成的内容可能不正确。图5-1 记事本主界面 图5-2 文件保存页面

# 六、问题分析与解决

# 首先是对ui问题的解决：在开发时我们忽略了对窗口最小值的限制，导致某些情况下ui会极为奇怪。对此，我们直接使用setMinimumSize(600, 400);对最小尺寸进行了约定。

# 其次是右键问题。在尝试了很多种自己写的方式失败后，偶然发现Qt这玩意自带右键菜单，因此我们直接删除了自己的相关代码以求精简，以及确保稳定性。

# 再次，我们打算将其作为一个开源的、长期维护的项目，并在后续逐渐推出类似于Typora“所见即所得”的Markdown编辑器模式，可以看到我们在扩展功能中预留了这个下拉项。不过在权衡时间以及学习成本的情况下，暂作为后续版本的计划进行处理。因此该应用的部分工具无法正常点开查看，基本都是由于这个原因。

# 还有乱码问题。在检测过程中，当选择打开docx或doc后缀名文件后会出现乱码问题，初步判断是编码问题，因本人知识面限制，暂时未找到两全其美的方法，等待后续。

# 七、总结

通过该项目的实践，我逐步了解了Qt的基本用法，以及对数据结构与Qt的综合认识。