《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。

3、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。

二、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。

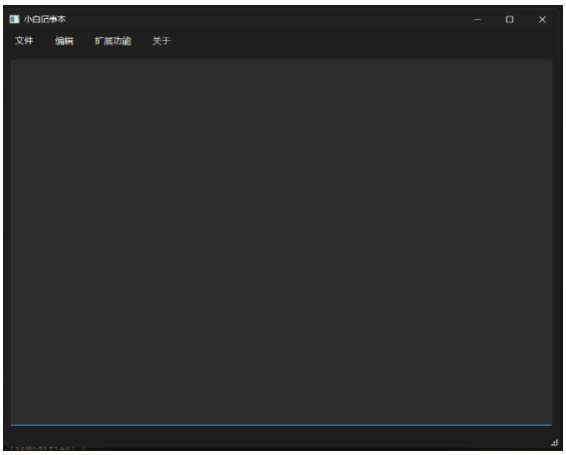

图 5-1 记事本主界面

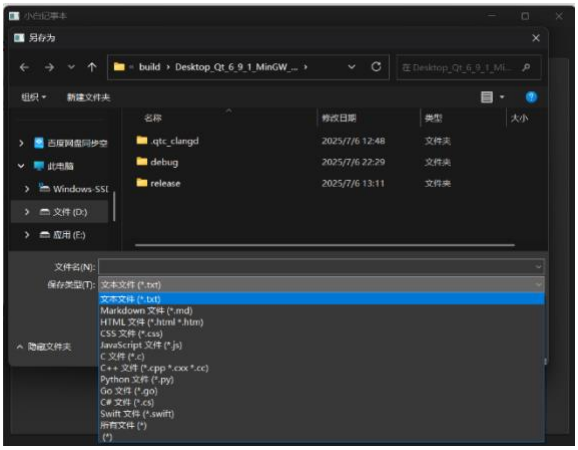

图 5-2 文件保存页面

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。