浙江大学

项目设计说明

课程名称: 面向对象程序设计

项　　目: 文本编辑器

姓名学号: 许子绎 3210104838

陈科 3210104844

王励勋 3210101760

指导老师: 陈奇

2023 年 6 月 18 日

目录

I 需求分析 3
1. 要求 3
2. 功能 3

II 总体设计 3
1. 代码结构设计 3
2. 实现关键点 3
2.1 设计的面向对象 3
2.2 界面 ui 设计 4
2.3 保存文档的数据结构 4
2.4 文本格式设置 4
2.5 搜索迭代器 5
2.6 文件列表 5
3. 数据主要保存格式类 NodeData 6
3.1 primate 有关函数 6
3.2 content 有关函数 6
3.3 primate content TextEdit 的转换函数 6
3.4 其余函数与格式的设置有关 6

III 系统模块说明 & 使用说明 6
1. 界面 6
2. 文本基本操作 7
3. 文本格式操作 7

IV 设计难点与解决方案 8
1. Qt 学习与使用问题 8
2. 文本格式的保存 8
3. 文本搜索实现 8

V 总结 & 感想 8
1. 开发经验 8
2. 程序不足 8

A 项目开发日志 9
1. 开发准备 9
1.1 环境配置 9
1.2 合作方式 9
1.3 文档写著 9

2. 项目搭建 9
2.1 文本编辑器框架搭建 9
2.2 功能实现 9
2.2.1 文本编辑基本功能： 9
2.2.2 文本保存与打开功能： 9
2.2.3 文字格式设置功能： 9
2.2.4 查找与替换功能： 9
3. 优化改进及后期总结 9
3.1 优化功能 9
3.2 添加了部分快捷键 10
4. Release 版本发布 10

I 需求分析

1. 要求

实现文本编辑器

2. 功能

1) 支持基本的文本编辑功能，包括输入、回退、插入、删除、撤销、重做等
2) 支持文件的保存和打开
3) 支持搜索与替换
4) 支持文字格式的设置，包括大小、字体、颜色等
5) 支持排版设置，包括对齐方式、自动分页等

II 总体设计

1. 代码结构设计

使用 Qt6.5.0 配合 Qt Creator 生成标准结构的项目代码。使用 CMakeLists 编译，build 文件夹内为本地构建项目目录，src 文件夹内放置源文件和头文件、贴图和格式文件。其中 images.qrc 文件能够读取 image 文件夹内的贴图 (jpg,png,etc.)，styles.qrc 能够读取 styles 文件夹内 css 格式文件。

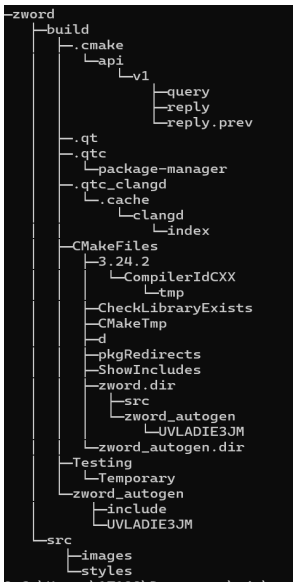

图 1: 项目文件目录

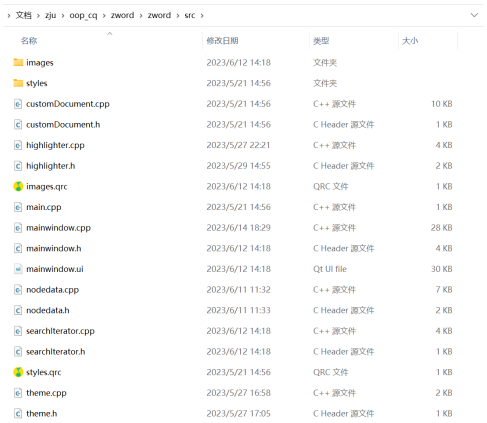

图 2: src 目录下文件

2. 实现关键点

2.1 设计的面向对象

我们在设计该文本编辑器的最初就希望构建一个能够一次性打开多个在不同文件夹中的笔记文本并进行编辑的。该文本编辑器意图面向希望能够快速进行便签记录，并对不同便签信息进行简单格式编辑和更改，做快速存储功能。

2.2 界面 ui 设计

我们使用了 Qt 自带的 ui 格式进行设置，只需要进行基本模块的拖动以及模块之间位置和格式关系的设置即可显示页面。在 mainwindow.ui 中可以进行操作，操作能够自动通过正则表达式存储。

Qt 中我们主要用到的模块有按键 (Buttons)、标签 (Label)、ListView 和 TextEdit。其中 TextEdit 能够通过 Qt 自带的提升操作，自动在 ui 文件中设置自定义的派生类（我们派生为 CustomDocument）。而主要运用到的格式关系有 Spacers 和 Layouts，能够分别提供模块的格式拉升功能和多个模块的排版。下图为 mainwindow.ui 的主要界面设计，可以注意到中间为我们设计的界面，右侧则为我们所加入的各种模块及其之间的关系。

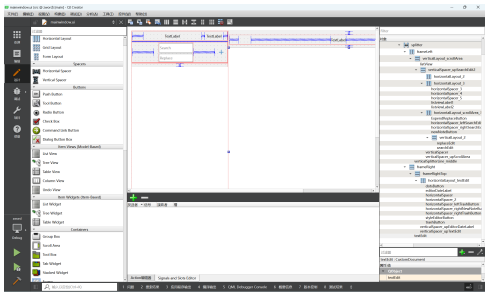

图 3: 界面设计：mainwindow.ui

2.3 保存文档的数据结构

由于我们保存的格式为.txt 纯文本格式，因此需要将格式保存为文本，类似 markdown，因此最终考虑将文档保存为带格式原始文本和显示文本两种。其中带格式文本为全文本存储；显示文本为每一句一个字符串，并使用一个 vector 存储。

在每次打开文件和保存到本地文件的过程中，带格式文本进行更新，这个过程对长文本的处理相对较漫长。但当每次编辑时，显示文本能够实时更新，而且更新方式为：找到更改的文本段对应行字符串进行更改，如果添加和删除字符串再对后续文本段依次更新。这样处理文本的方式能够更加有效地处理长文本。

在源代码中定义了 NodeData 类，该类能够实现上述文本数据结构的构建。

文本的显示、输入等的用户交互界面我们主要使用 Qt 的 QTextEdit 进行。并以 QTextEdit 作为基类，派生出 CustomDocument 子类，进行如超链接等更多模块的拓展。

2.4 文本格式设置

由于主要的文本编辑显示窗口使用 QTextEdit 编辑，Qt 中已经有很多非常基础的文本格式设置了，但这些格式的设置一来非常底层，二来只能够在显示的时候进行更改，不能保存到本地文件中，因此无法真正做到文本编辑器的功能。

考虑到我们保存为纯文本格式 (txt)，因此在保存文本格式的时候，可以借鉴 Markdown 一样的格式设置，通过特殊的符号和显示格式来实现简单格式的设置。

编译过程通过保存按键完成。

主要有以下三类文本格式：

- 加粗和斜体格式。

与 Markdown 的思路一样，通过在需要加粗和显示斜体文字前后加上相应符号。

- 文本总的字体、字号和颜色格式。

我们设定为：同一个文本文件使用同一个字体、字号和颜色。字体、字号和颜色的格式放在每个文件的开头，以键值对字符串的方式存放，需要时通过读取字符串并进行对应格式转换实现。

- 段落对齐格式。

在每个段落结束加上一个段落对齐的特殊符号以表示段落对齐格式。

2.5 搜索迭代器

没有选择直接调用 Qt 自带的搜索函数，而是自己实现了一个搜索迭代器。其内部实现使用了 KMP 算法。使用了一个名为 SearchIterator 的类，用于在文本中查找指定的字符串。该类包含以下公共成员函数：

- SearchIterator()：默认构造函数，初始化成员变量。
- SearchIterator(QString& search, QTextDocument* doc, int pos)：构造函数，使用指定的搜索字符串、文本文档和起始位置初始化成员变量。
- void Init()：初始化成员变量，将搜索位置和匹配位置都设置为 -1。
- ~SearchIterator()：析构函数，释放 next 数组的内存。
- SearchIterator& operator=(const SearchIterator& other)：赋值运算符重载函数，用于将一个 SearchIterator 对象赋值给另一个对象。
- bool operator++()：前缀自增运算符重载函数，用于查找下一个匹配字符串，返回值是为了区分末尾时匹配到字符的跳出与未匹配到字符的跳出，以此处理循环匹配。
- QTextCursor operator*()：解引用运算符重载函数，返回当前匹配字符串的 QTextCursor 对象。
- bool operator==(const SearchIterator& other)：相等运算符重载函数，用于比较两个 SearchIterator 对象是否相等。
- bool operator!=(const SearchIterator& other)：不等运算符重载函数，用于比较两个 SearchIterator 对象是否不相等。
- bool operator==(const QString& other)：相等运算符重载函数，用于比较当前搜索字符串是否等于指定的字符串。
- bool operator!=(const QString& other)：不等运算符重载函数，用于比较当前搜索字符串是否不等于指定的字符串。
- int GetPos()：返回当前匹配字符串的起始位置。
- bool IsEnd()：判断是否已经搜索到文本末尾。
- bool IsHit()：判断是否已经匹配到字符串，只要匹配到了字符串才会进行循环搜索。
- QString operator[](int pos)：重载下标运算符，用于获取指定位置的字符。

此外，该类还包含了一些私有成员函数和变量，用于实现字符串匹配算法。其中，next 数组用于存储字符串匹配时的跳转表，text_pos 和 pattern_pos 分别表示当前搜索位置和匹配位置，text_len 和 pattern_len 分别表示文本长度和搜索字符串长度，cursor 表示当前搜索位置的 QTextCursor 对象，pattern 表示搜索字符串，hit_at 表示匹配位置。

其内部使用了 KMP 字符串匹配算法，用于在一个文本串中查找一个模式串的出现位置。它的核心思想是利用已知信息来避免无效的比较，从而提高匹配效率。KMP 算法的实现基于一个跳转表 (也称为失配函数或 next 数组)，用于存储模式串中每个位置的最长公共前后缀长度，此表会在初始化时构建。KMP 的时间复杂度为 O(m+n)，复杂度较低，尤其是在模式串较长时，效果更加明显。

2.6 文件列表

由于考虑到希望设计成一个较为简约的文本编辑器，我们一开始没有考虑设计分页功能，转而将分页功能转为类似 vscode 的分文件功能。

左侧的文件列表通过点击能够切换不同文件。主要需要设计一个文件列表向量存储不同打开的文件。我们程序中使用了 vector<NodeData*> 变量存储所有列表，并设计 NodeData* 类型变量指向当前编辑文件。

其实无论是分文件还是分页功能，都是一个记录和清空的实现，再通过按键的设置储存住该记录即可。

3. 数据主要保存格式类 NodeData

3.1 primate 有关函数

- QString primate() const;
- void setPrimate(const QString& primate);
- void appendPrimate(const QString& append, int pos);

实现了带格式字符的纯文本 primate 的相关操作。

3.2 content 有关函数

- vector<QString> vcontent();
- QString rowContent(int row);
- void setRowContent(int row, const QString& content);

实现了显示在 TextEdit 上，不含文本格式字符的纯文本 content 的相关操作。content 使用 vector<string> 的格式进行存储和提取。

3.3 primate content TextEdit 的转换函数

由于三者的转换关系涉及文本格式的添加和删除，因此是比较繁琐的一环。

- void PrimateToContent();
- void TextEditToPrimate(CustomDocument * textEdit);

3.4 其余函数与格式的设置有关

III 系统模块说明 & 使用说明

1. 界面

界面总览如下：

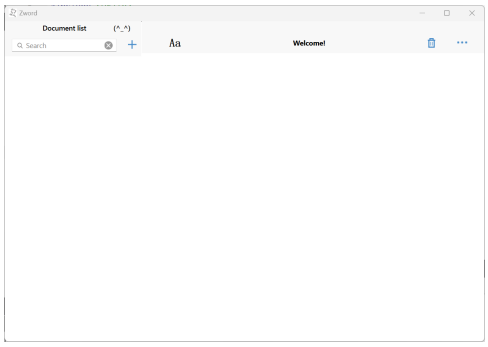

图 4: 空白界面

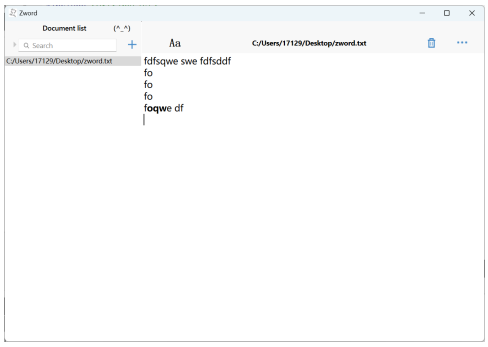

图 5: 加入文本界面

主要的界面分为两个部分：左边是打开的文件列表，右边是文本输入区域

使用时，可以一次性打开多个文本文件，文件将会依次显示在左侧文件列表中，并通过鼠标单击的方式能够切换显示的不同文件。

右侧的文本输入区域能够进行文本编辑。十分推荐使用者在已有的文本或是新建文本上进行编辑操作。

上面的栏目为功能栏。其中，左上角的 Search 主要实现查找和替换功能模块，单击 Search 左侧的小箭头可以打开替换栏。功能栏上方有各种功能按键，其中右上角居中位置的字代表了当前工作中的文件名。

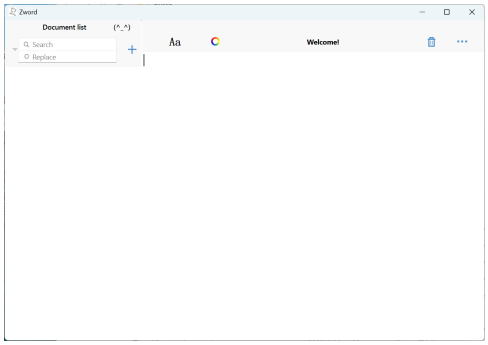

图 6: 搜索替换框的总界面展示

2. 文本基本操作

- 打开文本使用快捷键 Ctrl+O，只能打开已有文本
- 新建文本使用左上角的‘+’号，或者使用快捷键 Ctrl+N
- 保存文本使用快捷键 Ctrl+S
- 同其他文本编辑器一样，无法删除本地文本文件。若要删除当前打开的文本，可以点击右上角垃圾桶按钮或者使用快捷键 Ctrl+D
- 文本的撤销使用快捷键 Ctrl+Z，或者右键文本区选择 Undo
- 文本的重做使用快捷键 Ctrl+Y，或者右键文本区选择 Redo
- 文本的复制使用快捷键 Ctrl+C，或者右键文本区选择 Copy
- 文本的粘贴使用快捷键 Ctrl+V，或者右键文本区选择 Paste
- 文本的全选使用快捷键 Ctrl+A，或者右键文本区选择 Select All
- 文本搜索使用快捷键 Ctrl+F，或左上角的搜索按钮
- 文本替换使用快捷键 Ctrl+H，或左上角展开后的替换按钮。当然，也可以在搜索到黄色高亮文本后，按 Enter 键，即可实现替换。
- 文本的格式与颜色设置分别使用中间的两个按钮，为了实现的简单性，这两个按钮只能对整个文本进行设置，无法对文本的某一部分进行设置。

3. 文本格式操作

- 在文字的前后加上‘#’号，再使用 Ctrl+S 进行保存即可使‘#’中间文本加粗。（带‘#’的格式会保存到 txt 文本文件中，再次打开依然会显示该格式）
- 在文字的前后加上‘~’号，再使用 Ctrl+S 进行保存即可使‘~’中间文本斜体。（带‘~’的格式会保存到 txt 文本文件中，再次打开依然会显示该格式）
- c++ 关键字能够高亮显示（格式不会保存到 txt 文本文件中）
- 对网址（正确 url 格式网址）按住 Ctrl 键点击，能够使用默认浏览器访问该网址
- txt 原始文本开始进行了字体等格式的设置
- 使用中间的‘Aa’按键可以选择文本字体和字体大小，选择的字体和大小会保存到 txt 文本文件中，再次打开也会显示该字体，但格式操作不会有效果。
- ‘Aa’旁边的按键可以设置文本的颜色。
- 文本设置为一行最多 64 个字符，如果调整窗口大小，可以看到文本能够自动缩进
- 使用最右上角的三个点按键，能够清除当前文本的所有格式，但只能在打开文本时点击。
- Ctrl+L Ctrl+E Ctrl+R 三个快捷键能够进行段落的左、中、右对齐操作。其中，如果没有选中文本，将对所有文本段落进行对齐操作；如果选中文本，将对文本所在段进行对齐操作。保存后，段落对齐格式

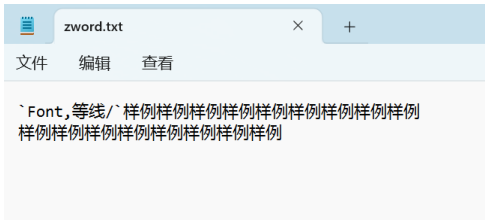

图 7: 开始进行字体格式设置

能够保存到 txt 文本文件中，再次打开也会显示该文本对齐格式。

IV 设计难点与解决方案

1. Qt 学习与使用问题

本次项目最大的难点还是在于 Qt 的学习和使用。由于 Qt 为使用者搭建了一个非常完整但同时很基本的开发环境，因此在实现文本编辑器的过程中，绝大时间都花费在通过官网和各类资源查找 Qt 某个具体函数的使用方法。而项目代码中的很大一部分都在调用 Qt 自带函数、重载或 override 其中某个基本函数等。

Qt 的 ui 搭建使用的是 Qt 自带的 ui 设计器，因此在实现过程中，需要在 ui 设计器中添加各类控件，然后在代码中调用这些控件。这是首次搭建 UI，使用的并不熟练，在过程中做了很多无用功。

2. 文本格式的保存

另一个难点是文本格式的保存。因为需要处理带格式文本、不带格式纯文本和显示格式文本三类文本，因此在三类文本的相互转换和同步十分关键。而这里格式的显示和保存是一个双向的过程，既要实现打开一个 txt 文本文件能够显示格式，又要实现在文本编辑器中更改了文本格式后能够保存到 txt 文本文件中。

在具体的实现过程中，随着功能的增加和一开始结构设置的不完善，出现了很多代码的冗余，这也给后续功能的改进和实现产生了较大的负担。

此外，由于许多编辑器的功能是基于已保存的文本实现的，因此在实现过程中，需要注意在文本未保存时的各类操作的正确性。

3. 文本搜索实现

在实现过程中，由于 KMP 算法的特殊性，需要对文本进行预处理，因此需要注意迭代器的构造时 next 数字的构造与析构时的正确性。

V 总结 & 感想

1. 开发经验

- 认识到 Qt 的强大，第一次深入了解 Qt
- 交互程序 debug 的不容易
- 完整的开发经验，Github 仓库、Qt 程序开发、Latex 文本开发、Markdown 中期程序说明、Debug 和 Release 版本的发放、开发日志的完整记录。

2. 程序不足

- 仍然有一些 bug 值得完善。
- 对文字字体和颜色的更改仅处理了全部文字（所有文字的更改格式记录会比较麻烦）
- 如果时间充沛，可以考虑引入 sql 数据库等进行更丰富的保存。
- 实现了分文件功能，但没有实现分页功能

A 项目开发日志

1. 开发准备

4 月 25 日到 5 月 6 日

1.1 环境配置

于 4 月 25 日到 5 月 6 日完成了 Qt 装载以及环境的配置。使用 Qt6.5.0 配合 Qt Creator 或 VS2022 作为开发环境，配置 VS + Qt 的参考教程。Qt 为跨平台 C++ 图形用户界面应用程序开发框架。

1.2 合作方式

于同期使用 github 开设私人仓库并设置合作者实现合作开发，可能使用 PR (Pull Requset) 实现更好的整合。

Github 仓库. 该仓库将于验收当天开源 Public。

1.3 文档写著

于同期在在线 LaTeX 编辑器网站 Overleaf 创建了文档，在线共享编辑。

2. 项目搭建

5 月 7 日到 6 月 10 日

2.1 文本编辑器框架搭建

5 月 7 日到 5 月 13 日使用了 Qt 自带的 ui 格式进行设计，通过基本模块拖动并安排其位置关系进行了页面的设计，完成了文本编辑器基本框架的搭建。

2.2 功能实现

5 月 14 日到 5 月 20 日

2.2.1 文本编辑基本功能：利用部分 Qt 自带函数与自己编写的函数，实现了文字输入，回退，插入，删除，撤销，重做等文本编辑的基本功能

5 月 21 日到 5 月 25 日

2.2.2 文本保存与打开功能：对文本编辑器中的文件进行了两类方式保存，保存带有格式标识的显示文本以及最终的显示文本，格式均为 txt。

5 月 26 日到 5 月 31 日

2.2.3 文字格式设置功能：为使显示文本格式能够在保存文件中显示，我们借鉴了 Markdown 中应用特殊符号设置的显示格式实现字体格式的设置，实现了加粗与斜体格式，字体、字号与颜色格式，段落对齐格式三方面的文本格式设置。

6 月 1 日到 6 月 10 日

2.2.4 查找与替换功能：使用了 Knuth-Morris-Pratt 字符串查找算法，自己编写了搜索迭代器，实现了文本的搜索；同时对查找到的文字，实现调用下方的 replace 进行替换。

3. 优化改进及后期总结

6 月 11 日到 6 月 18 日

3.1 优化功能

添加了文档排版设置，同时进行了贴图

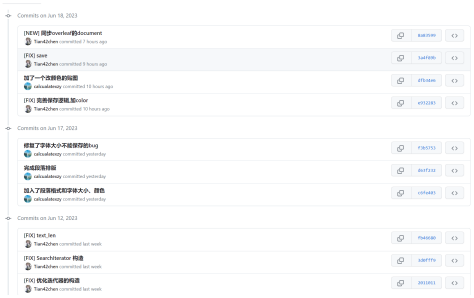

图 8: Github 上 6 月部分 commit

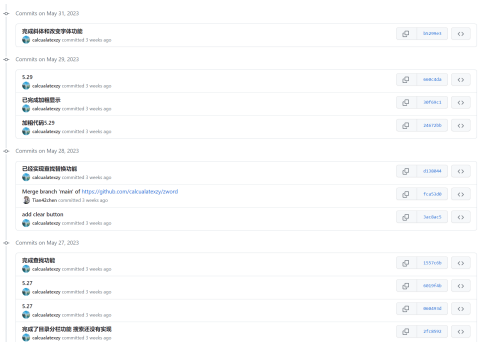

图 9: Github 上 5 月部分 commit

3.2 添加了部分快捷键

扩展实现了键盘与 Qt 界面的交互，编写了部分按键函数，添加了 ctrl+o：打开文本；ctrl+n：新建文本；ctrl+z：撤销文本等快捷键，使得快捷键功能基本与主流编辑器相符，使用更方便。

4. Release 版本发布

Qt 使用 Release 编辑完后，再使用 windeployqt 发布 Release 版本文件。并将版本文件发布至 Github 上。