

高级程序设计 第二阶段课程项目报告

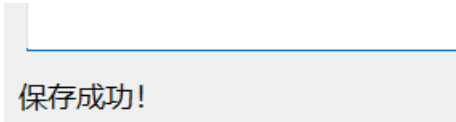
201220115 杨青云

关于选题

在本项目中，实现了一个具有不完善功能的文档预览器，它支持markdown到html的解析，并将解析后的html渲染到预览框；支持不完善的Latex和Word预览功能。支持对.md和.tex文件进行编辑并实时预览，支持html语法。

功能详述

提供保存文件和打开文件的功能，并利用文件后缀判断文件类型，进行不同语言的解析；



保存成功!

图：保存成功输出提示文字

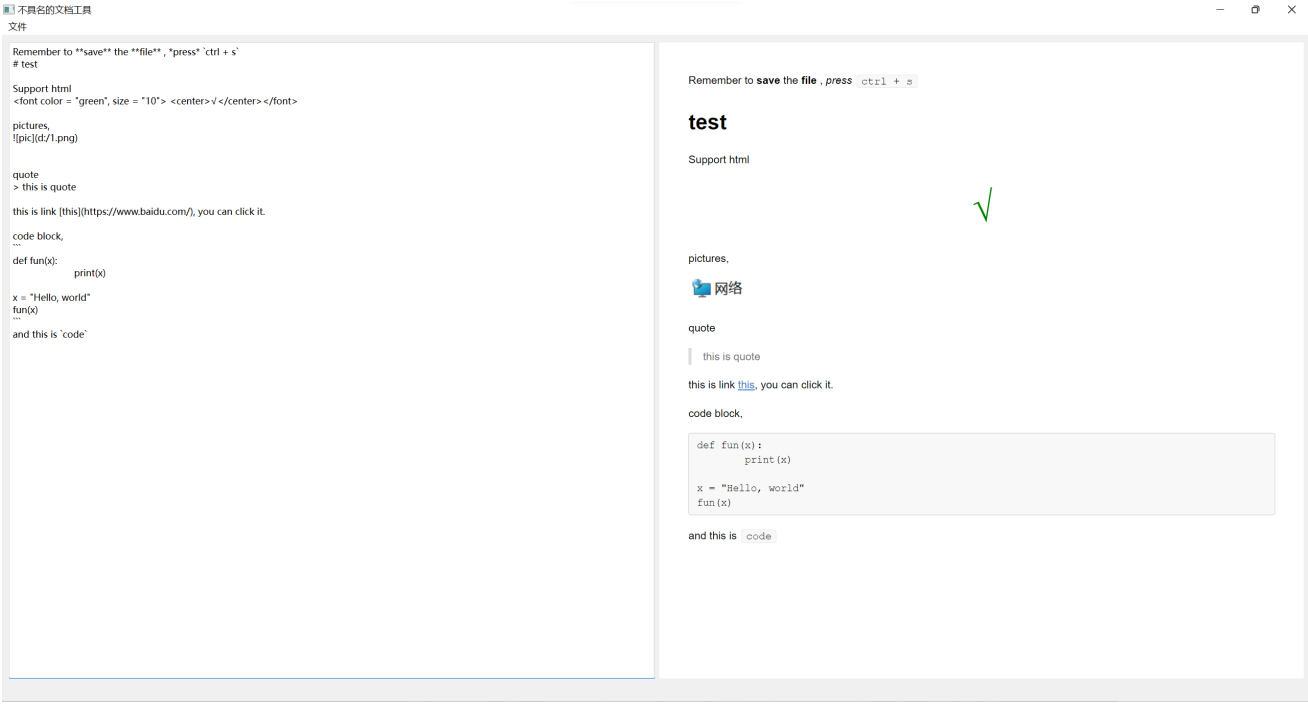
提供一个编辑框和一个预览框，用户可以任意更改编辑框中的文本，预览框会实时显示编辑后的效果；

当用户利用菜单栏中的打开文件对话框打开一个文件后，文件内容会被显示到编辑框中，并在预览框中展示渲染后的效果；

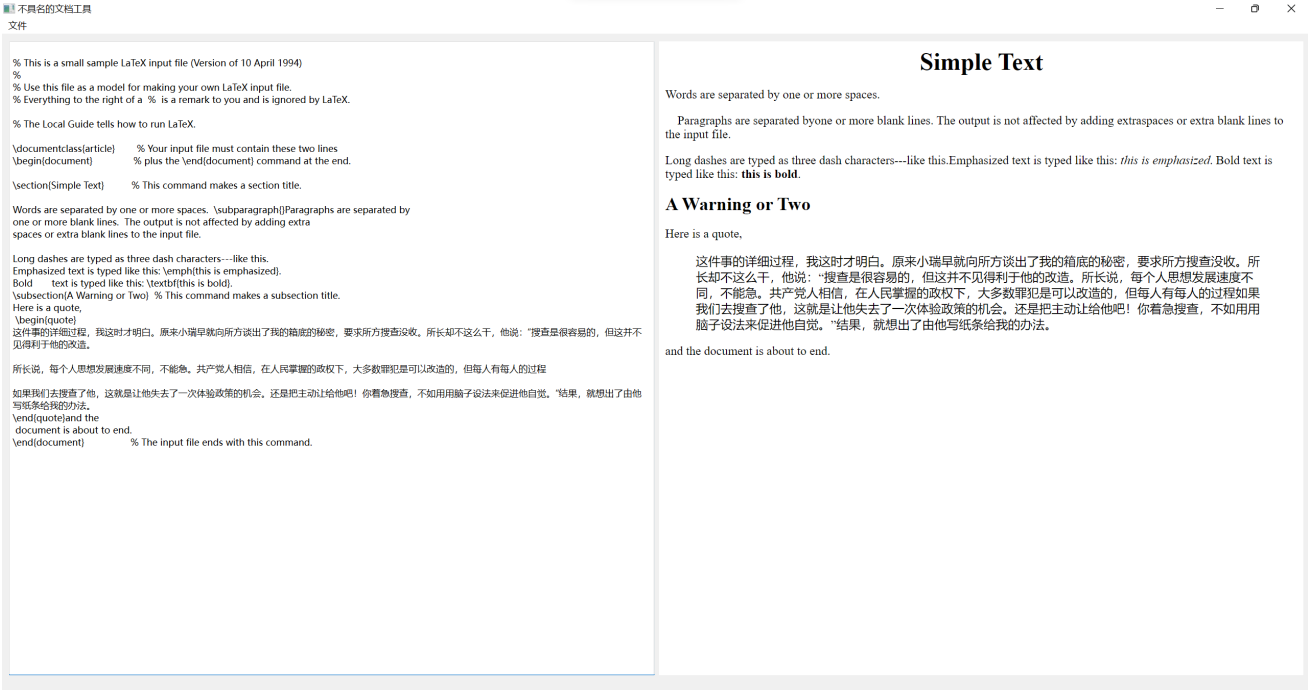
在未打开文件的编辑框编辑时，会提供html语法的实时渲染；

1. 会在预览框中展示渲染后的结果；
2. 支持**部分markdown常用语法**（包括 6级标题，空行开启新段落，行末多于两个空格换行，粗体，斜体，列表，嵌套列表，引用，嵌套引用，行内代码，代码块，链接，图片，HTML标签）和**部分Latex常用语法**（包括标题、作者、日期、三级小标题、段落、子段落、引用、粗体、斜体、注释行和注释块等）以及**Word文档内容**的查看
3. markdown文本默认提供一个.css主题文件（注：默认markdown主题来自互联网¹），用户可以使用自己喜欢的主题文件替换该主题；
4. 内嵌的web浏览器，可直接打开文本中的链接

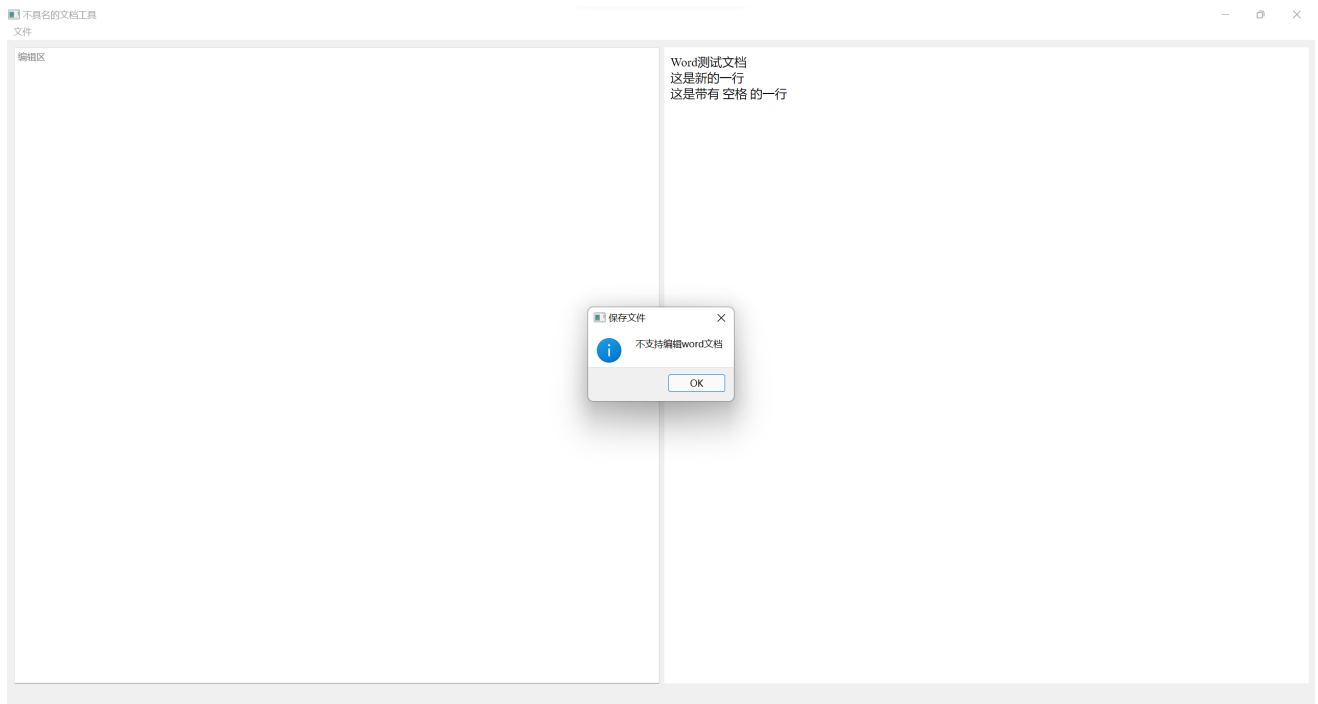
最终效果如下，



图：markdown



图：Latex



图：Word以及尝试保存Word文档时的提示

实现思路

编辑和预览功能：检测到编辑框文本发生变化时，会发送信号给预览模块，预览模块获取编辑框全部文本后根据文件类型进入不同的解析模块，解析模块根据得到的文本进行解析，并输出符合html语法的文本，预览模块将该文本写入html文件，并在预览框中载入该html文件，从而实现实时预览。

打开文件功能：利用Qt的文件读写接口，将打开的文件的全部内容写入编辑框，实现文件打开功能。

保存功能：利用Qt的文件读写接口，将当前编辑框的内容全部写入文件，实现保存功能。

markdown到html的语法解析：

首先建立markdown的语法树结构，语法树从根节点开始，节点结构如下，

```
struct node
{
    int type;
    std::vector<node *> children;
    QString text[2]; //0-plainText 1-url,path,...

    node(int t):type(t){}
};
```

其中，type表示该节点的类型，类型为markdown语法要求的几种特殊结构（标题、段落、图片、链接等）或纯文本；

children是一个容器，用于存放该节点的子语法结构，如：在标题中的斜体内容即为标题的子节点；

text是一个存有两个字符串的数组，其中text[0]存放该节点的文本内容（如：标题最终显示的文字），text[1]存放该节点的链接内容（如果有，如：图片的路径，网页的地址）；

每一个节点即为一个token，定义根节点的类型为纯文本，但不存储文本信息，其children为段落或标题，即从总体来看，文档是由一个个段落和标题组成的。

其次是生成语法树的过程，对于markdown，可以大致将关键字分为两种：行首关键字和行间关键字，因此将解析过程整体分为行解析和行间解析，行解析读取每行的开头，并初步判断该行的类型，如标题、代码块或列表等，为它们生成节点，然后进行行间解析，特别的，空行只生成一个新的段落，然后读取下一行，不进行行间解析。行间解析将会对传给它的字符串进行逐字符的解析，识别其中诸如粗体、斜体的结构并为之生成节点附在上层节点上。

生成语法树后，只需要对其进行深度优先遍历，即可解读出其中的内容，具体到解读为html，设置两个全局数组分别存储不同类型的html标签头和标签尾，在遇到节点时，依照其类型取出对应的标签头，然后依html语法拼接文本、链接和标签尾即可得到html文本。

当然，对于图片和链接，因为它们的结构特殊，因此需要进行特判转换为html；而对于换行而不换段落的行为，因为
标签无标签尾，因此直接在解析行时添加换行标签而不是等到语法树遍历时进行转换。

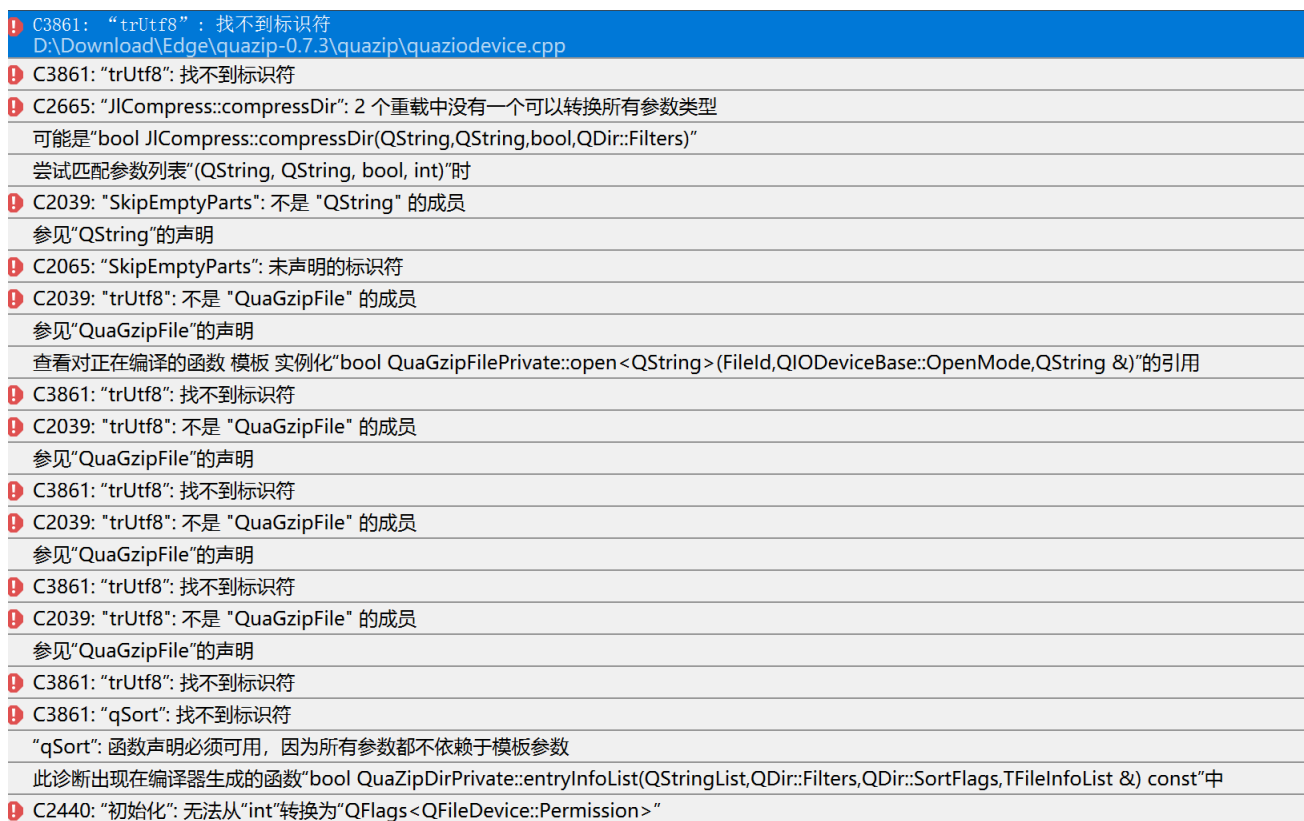
Latex到html的语法解析：

Latex语法多以keyword{words}的形式出现，因此逐个字符读入并进行解读，对于每个字符，查看其和其后的字符是否形成关键字，并依照不同关键字对大括号内文字进行不同处理。特别的，对于换行\\、\newline、换段\par、\n\n、单行注释%，进行特殊处理，例如单行注释则忽略其后的字符直至行尾，换行直接向html输出一个
等。

对于段落，使用一个bool型变量记录当前是否为与某个段落，并以此决定换段时输出</p><p>还是<p>。

Word文档的预览

遇到的第一个问题是解压缩库，由于Qt本身不提供解压缩库，因此需要寻找第三方库，经过搜索，找到了许多人推荐的Quazip库，但是目前只支持到Qt5的版本，而由于Qt6以下版本的Qt对于中文支持并不够良好，因此项目是由Qt 6.2写的。因此遭遇到无法编译库的问题。



图：尝试编译库时的问题

看到这个情况，本来已经打算放弃，但是发现它们大多并不麻烦，在替换了这些不再被支持的旧函数之后就解决了问题。之后的情况就比较顺利，对文档中的document.xml进行解读即可。关于对document.xml的解读参照了互联网的讨论²。

其他

因为对markdown、Latex、html这类语言较为陌生，因此完成文本预览器的过程较为曲折。起初的想法是写一个大循环设置很多标志位进行解读，但是实际实现时发现比较困难，这种写法很容易写着写着不知所云且bug多。

然后查找相关资料了解到DOM树结构，并在互联网³找到了介绍markdown语法树结构的文章，参考了其中的语法树结构进行项目的实现。

借由这个项目，了解到以前所不了解的Latex这一优秀文档工具，并了解到一些以前不知道的markdown特性。

另外，意外发现Qt的文本编辑框QTextEdit类也提供markdown转html的接口，

	QString	<code>toHtml()</code> const
	QString	<code>toMarkdown(QTextDocument::MarkdownFeatures features = QTextDocument::MarkdownDialectGitHub)</code> const
	QString	<code>toPlainText()</code> const
void		<code>setHtml(const QString &text)</code>
void		<code>setMarkdown(const QString &markdown)</code>
void		<code>setPlainText(const QString &text)</code>

QTextEdit is an advanced WYSIWYG viewer/editor supporting rich text formatting using HTML-style tags, or Markdown format. It is optimized to handle large documents and to respond quickly to user input.

1. <https://github.com/jasonm23/markdown-css-themes> ↩
2. <https://blog.csdn.net/lmhuanying1012/article/details/78764041> ↩
3. <https://blog.csdn.net/Mmonster23/article/details/108600170> ↩