技术文档

20172131138 曾德明 4 班

————文本编辑器

一、 开发过程

1、设计过程

前期准备:

下载 QT5.8.0，安装搭建好对应的开发平台和相关软件，从官网上下载静态库文件 QScintilla_gpl-2.11.1，从网上下载对应功能的图标，如复制，黏贴，剪切，撤销等，同时了解 QT 编程相关的基础知识，了解相关菜单栏和 QT 信号与槽的机制

1.1、设计文本编辑器的主界面

首先用 QT 构建一个桌面 QT 应用项目，然后在新项目中建立 Qt source 文件，把需要用的照片的文本文件作为资源引入。本软件界面根据 notepad++软件的布局来实现，软件的主窗口中间添加入一个分页窗口作为文本编辑区域，上部为菜单栏，设有文件，编辑等日常的菜单，工具栏则有复制，黏贴等常用操作，底栏则显示相关的文件信息。

1.2、设计文本编辑区域的显示界面

文本编辑区域的最左边要显示行号，同时光标所处的行会显示灰色。

1.3、实现菜单的细化功能

文本编辑器的菜单栏中，文件菜单包含着新建，保存，另存为，打开等操作，编辑菜单包括撤销，恢复，复制，黏贴等操作，以此来方便文件的打开和编辑，同时为保存和打开设置对应的弹窗显示效果。然后将相应的按钮点击事件与函数进行绑定，实现点击操作。

1.4、添加对应语言处理功能

在菜单栏加入语言菜单，里面添加 c++，Java，python，normal 四种选择，选择每一种则可对应进行语法高亮和关键字补全，同时保存时可以根据对应格式进行保存。

1.5、关闭界面

分页窗口设置为可关闭，但如果主界面只有一个分页窗口就不可关闭，运用快捷键 ctrl+q 或者直接点击右上角的 x 即可直接关闭软件。

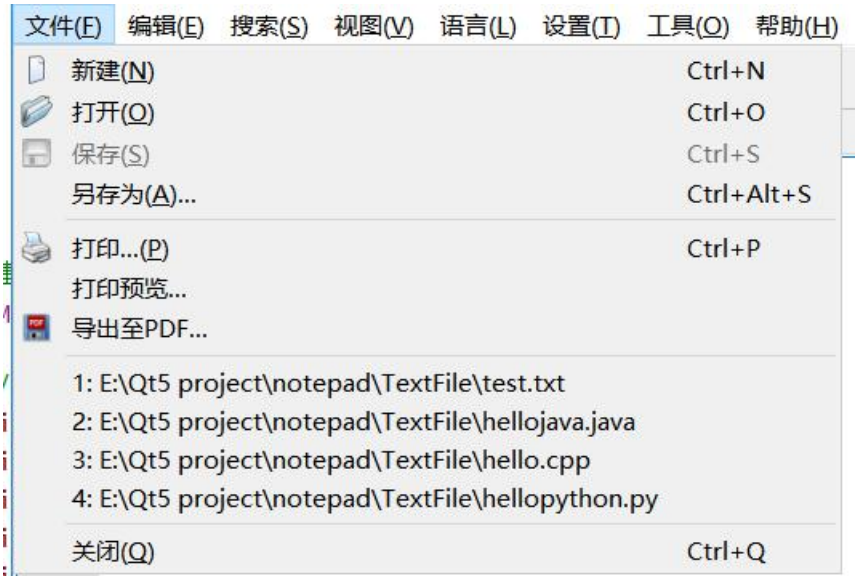
2、实现过程

2.1、设计文本编辑器的主界面

首先新建一个桌面 QT 项目，然后新建 images.qrc 资源文件把需要引用的图标加入到项目中。在 mainwindow.h 中定义显示窗口类 Mainwindow，在里面添加分页窗口 QTabWidget 类，菜单栏 QMenu 类，工具栏 QToolBar 类，和自定义的 QTextEditor 类，QStatusBar 是父类 QMainwindow 类自带的属性。

Mainwindow 类继承于 QT 的窗口类 QMainwindow 类，自动生成一个软件主窗口，this->resize(1024, 768)等基础函数用于设置窗口大小在主窗口中添加 QTabWidget 类对象 m_pTabWidget,同时把自定义的文本编辑器类对象 m_pTextEdit 使用相应的函数设置为分页窗口的中心窗口 this->setCentralWidget(m_pTabWidget)

createMenus()函数用于创建窗口的菜单栏，使用 QMenu 类中的方法如 menubar()->addMenu(“文件(&F)”)来进行菜单栏的添加，然后在 mainwindow.h 文件中添加对应的 QAction 如 newAction(新建动作)，然后用函数 addAction()把动作加入对应的菜单之中。



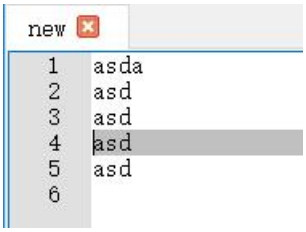
createToorBar()函数用于创建菜单栏之下的工具栏，主要用途是把常用的功能汇集到工具栏之上，减少了操作，通过调用 QToolBar 类中的 addAction()动作来把常用的动作添加入工具栏

createStatusBar()函数用于创建主窗口最下方的状态栏，以显示文件信息，换行模式，插入模式等基本信息，通过窗口父类的 statusBar()->addPermanentWidget 函数把描述信息用的字符串类 QLabel 加入最底部的状态栏

readSettings()函数与 writeSetting()函数相搭配来使用,writeSetting()的功能在于获取写入配置文件，记录最后一次打开时软件窗口的大小（用 saveGeometry()），然后把该函数放入 QT 窗口类继承函数 closeEvent 中，每次关闭窗口就写入配置，然后把 readSettings()函数放入构造函数中，每次打开软件就读取上一次的配置信息。

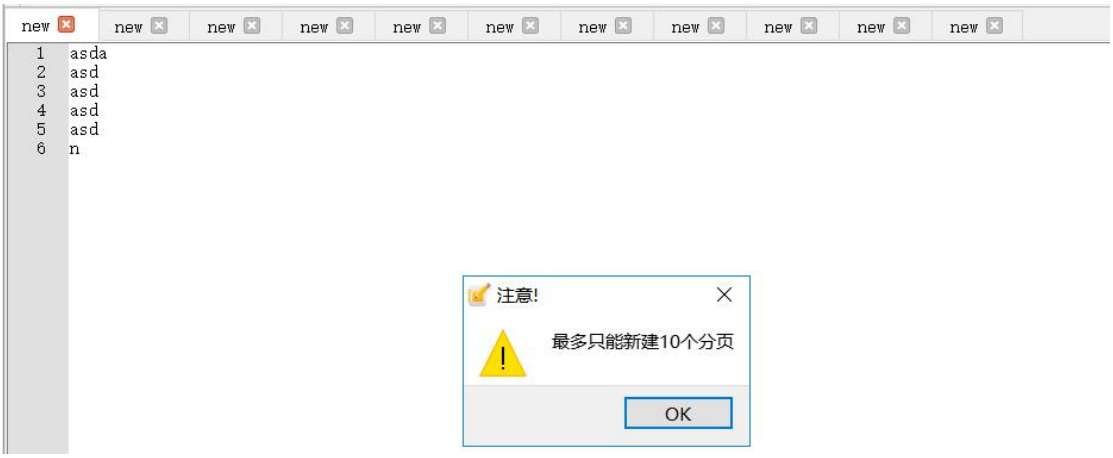
2.2、设计文本编辑区域的显示界面

文本编辑区域通过自定义一个文本编辑器类 QtextEditor 来进行编辑，继承于 QsciScintilla 类，文本编辑器类中主要的属性是判断不同系统的换行方式（Windows，mac，unix），以及文本的输入方式（正常 or 插入模式），然后通过把文本编辑器类的对象添加在主窗口类 mainwindow 中，在分页窗口 m_pTabWidget->addTab(m_pTextEdit, tr("new"))，把文本编辑框对象 m_pTextEdit 放入到分页窗口中，然后用 setMarginType 函数设置添加页边，setMarginLineNumbers 启用行号，setCaretLineBackgroundColor 设置选中行的背景色（Qt::lightGray），通过 showCursorPosition(int, int)函数与对象 m_pTextEdit 进行 connect，用于显示光标所在的位置。



2.3、实现菜单的细化功能

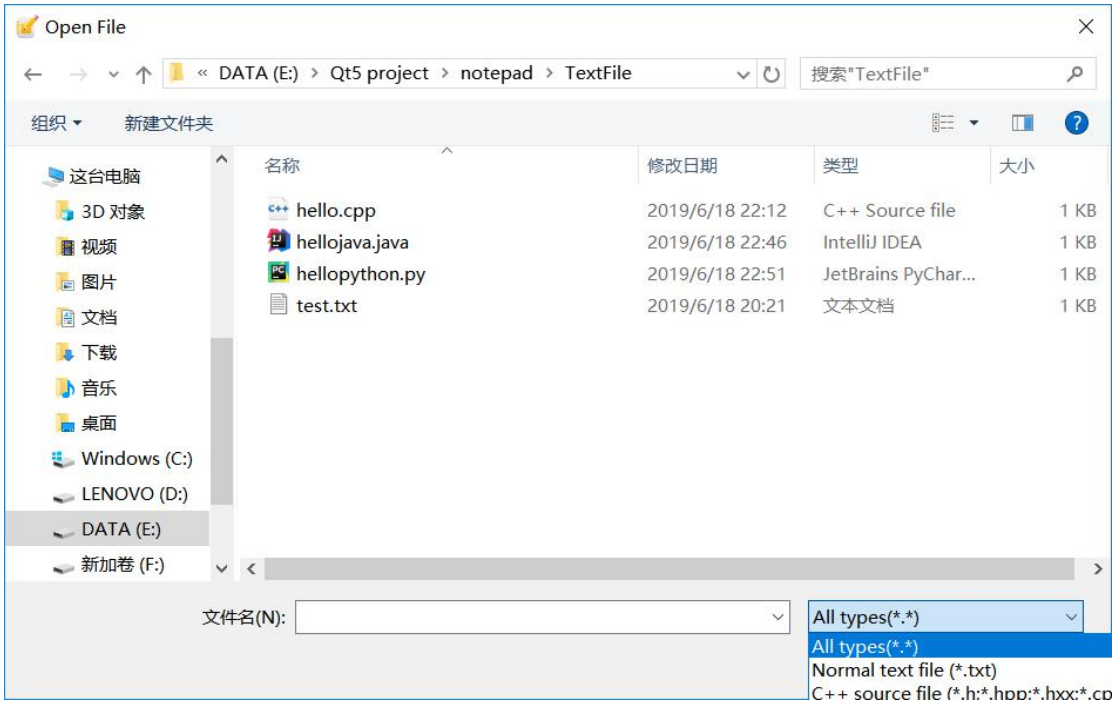
新建的实现通过函数 newFile()实现，其中的原理与原本的构造函数类似，通过创建一个新的文本框 NewTextEdit[i]，然后通过分页窗口类的 addTab()方式把新的文本框加入分页窗口，在界面上体现出来就是多出了一个分页，此时表示为逻辑上新建了一个文件，程序设置了一个 NewTextEdit 数组，最多能够同时新建 10 个文件（即是个分页窗口），当超过十个的时候则会有相应提示。



打开的实现通过 open()函数实现，点击打开时，激活 openAction 操作，函数 open()生成一个 QT 原生的提示框 QFileDialog，通过 getOpenFileName()函数设置打开初始路径".", 打开文件类型选择"All types(*.*);;……Python file (*.py;*.pyw)"。

保存与另存为相互结合使用，当点击保存时，激活 saveAction，调用 save()函数，在函数内，如果判断此文件为新文件(即从来没有创建过)，则调用 saveAs()函数执行另存为操作，如果文件为已有文件打开改变，则调用函数 saveFile()。saveAs()函数中，也类似 open()函数先弹出 QT 原生的提示框 QFileDialog，然后再调用 saveFile()函数。saveFile()函数中，首先使用输入输出流类 QTextStream 新建一个对象 out，然后通过

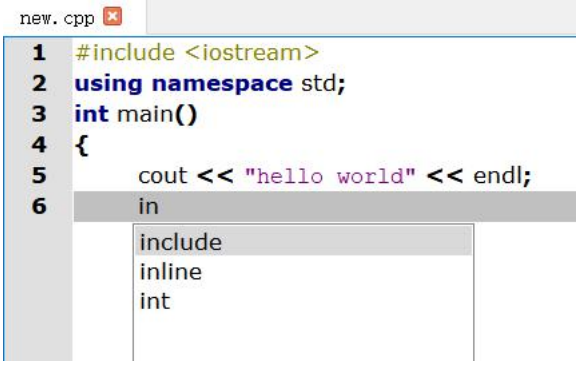
out<<m_pTextEdit->text()把文本保存，同时调用函数 setCurrentFile()来改变文件的名字和窗口的状态。



2.4、添加对应语言处理功能

语言功能除了原本的文本之外还设置了 c++，Java，python 等，一共有 4 种语言，所以设置一个数组 languageModeAction[4]来表示一共有四种语言变化动作，然后在引入头文件 Qsci/qscilexercpp.h，Qsci/qscilexerjava.h，Qsci/qscilexerpython.h。

把 languageModeAction 与 changeLanguageMode()作为信号和槽连接，前者通过鼠标点击后发出信号，调用后者对相应的分页窗口进行文本编辑的改变。在后者的操作中，首先通过 QAction *act=qobject_cast<QAction*>(sender())来获取动作的名字(如点击 c++，则 act->iconText()会返回字符串"c++")，然后根据动作的名字，设置不同的词法分析器（如 c++，则新建一个 QsciLexerCPP 类对象 textLexer），然后根据当前正在活跃的分页编号 m_pTabWidget->currentIndex()，给相应的文本编辑框添加上词法分析器 m_pTextEdit->setLexer(textLexer)，然后搜索对应语言相应的关键词，编写一个 txt 文件，把关键词一行一个写在内部，然后通过创建资源文件 keywords.qrc 来引入文本资源，接着在代码中添加 QsciAPIs *apis = new QsciAPIs(textLexer)（这个类运用之前新建的词法分析器进行构造生成），通过 load()函数把之前写的 txt 关键字文本读取保存进来，最后设自动补全提示 setAutoCompletionThreshold(1)



2.5、关闭界面

QT 窗口应用中，直接点击右上角的 x 本身会调用 closeEvent 进行关闭口操作，另一种方式，则是通过在文件菜单中增添一个动作 addAction(exitAction)，添加上快捷键 setShortcut(tr("Ctrl+Q"))，然后让动作与 QT 的 close()函数进行 connect，然后就可以直接使用快捷键进行窗口的关闭。

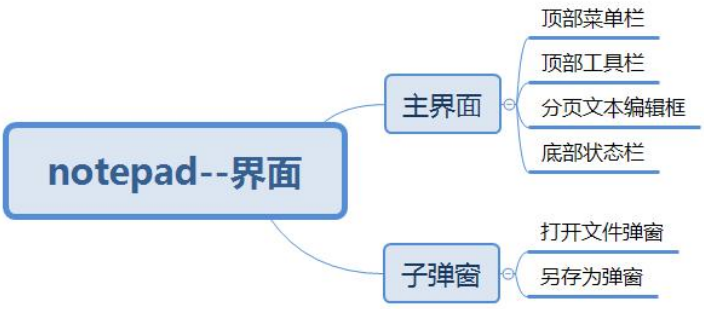
对于分页窗口来说,关闭一个分页窗口,就类似于把该分页移除,这里使用 connect,把分页窗口类和移除函数 removeSubTab(int)连接在一起,通过 QT 原生提供的 tab 关闭请求信号 tabCloseRequested(int)激活移除函数，同时向移除函数传递一个参数，即关闭的分页的标号（从 0 开始计算），removeSubTab(int)则通过把 NewTextEdit 数组中的对象往前移，覆盖需要删除的对象的位置，实现逻辑上的删除，同时把分页去掉。

二、 技术方案

1、功能组成

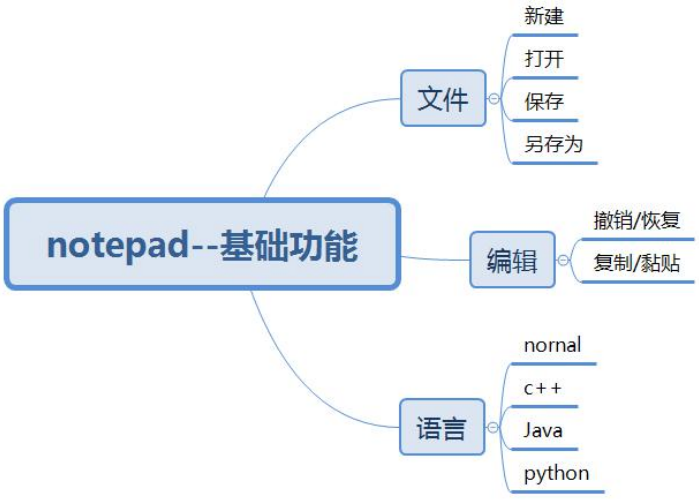
1.1、界面

界面包含一个主界面，两个子弹窗。



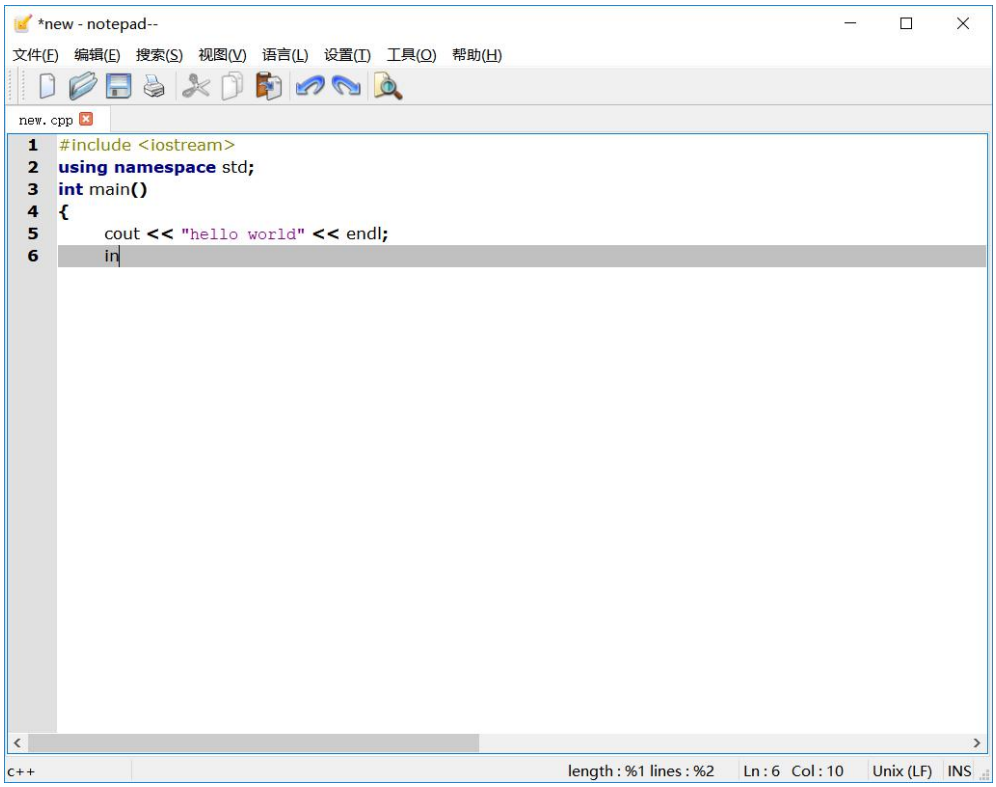
1.2、功能

新建，打开，保存，另存为，撤销，复制黏贴，语言模式等基础功能



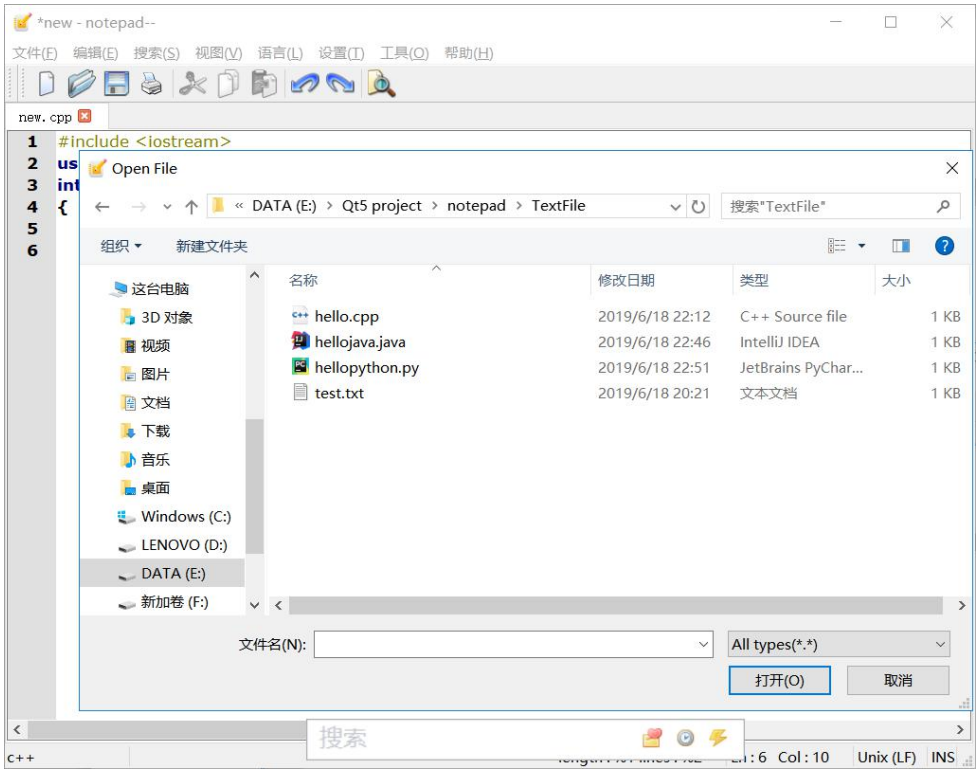
2、界面组成

2.1、主界面



主界面

2.2、打开&保存界面



打开&保存界面

3、 数据结构

1.1、

使用类对文本编辑框进行了初步封装（TextEditor 类），然后再在主窗口类中加入一个类属性来进行在窗口中添加文本编辑框。

1.2、

在创建多个分页窗口的过程中，使用数组保存每个分页窗口的对象。

1.3、

textchange()信号与函数 textEditorModified()进行 connect，如果文本编辑区域有变化（如输入了新文字），函数则会检测到，并且执行相应的操作（此处执行的操作是 saveAction->setEnabled(true)，即文本发生改变，允许执行保存动作）。

1.4、

无论是新建，打开，保存/另存为，加载文件，其中都用到了一个函数，即 void setCurrentFile(const QString &fileName, int i)，这个函数的作用是用于改变文件名字的显示，第二个参数是指分页窗口的编号。

三、 技术讨论

1、存在问题

1.1、

软件仅仅完成了初步的功能，如新建，打开等，其余的功能如文本查找，生成 MD5 值，打印为 PDF 等功能尚未完善。

1.2、

新建文件和保存文件方面，因为本人对分页窗口的信息显示不太熟练，所有会出现文件名改变了但是标题没改变的情况，其次，此时的新建也只是静态的通过数组来限定，灵活性不高，而且每个新窗口必须重新连接 action。

1.3、

语言模式方面，初步只能通过每种语言写一种词法分析器去添加，而且语言更换的动作被分成了数组的形式，若需要添加一种语言耗费时间较多，不灵活。

1.4、

代码中部分函数存在代码冗余稍高的情况，不同的功能的实现有若干部分的代码是一样或者相似的。

2、改进方向

1.1、

尝试完善文本查找功能和PDF打印功能。

1.2、

可以尝试更换数据结构，不使用数组，改用双向链表的方式来记录新生成的文本编辑框，这样就可以动态的进行增加和删除操作。同样的，如果有action需要多次的用到，也可以运用链表进行区分，每个结点包括自己的action

1.3、

语言模式上，尝试把每种语言的动作（c++，Java，python等）抽象成一个languageChangeAction，每次点击后产生的信号就只唤醒这个动作对应的语言模式改变函数，然后通过qobject_cast<QAction*>(sender())来获取action对应的名字(如c++，Java，python都对应languageChangeAction，但他们反解析后获得的字符串是不一样的），然后根据不同的语言设置相应的变化。

1.4、

可以尝试合并有冗余代码的函数，如果的确需要分开成两个功能进行操作，可以考虑添加一个新的参数来进行函数代码运行的选择（即如果是0则运行此段代码，为1则运行另一段……）。