目录

[Qt 作业1说明文档 2](#_Toc519378592)

[作业要求： 2](#_Toc519378593)

[使用说明： 2](#_Toc519378594)

[**1.** **主界面：** 2](#_Toc519378595)

[**2.** **关闭文件：** 2](#_Toc519378596)

[**3.** **新建文件：** 3](#_Toc519378597)

[**4.** **打开文件：** 5](#_Toc519378598)

[**5.** **保存文件：** 5](#_Toc519378599)

[**6.** **另存为：** 5](#_Toc519378600)

[**7.** **搜索：** 5](#_Toc519378601)

[实现思路： 6](#_Toc519378602)

[**1.** **文件保存实现方法：** 6](#_Toc519378603)

[**2.** **文件打开实现方法：** 7](#_Toc519378604)

[**3.** **弹出对话框：** 8](#_Toc519378605)

[**4.** **文档搜索：** 9](#_Toc519378606)

[走过的弯路： 10](#_Toc519378607)

*github账号：chendante*

*github项目地址：https://github.com/chendante/-txt-Qt.git*

# Qt 作业1说明文档

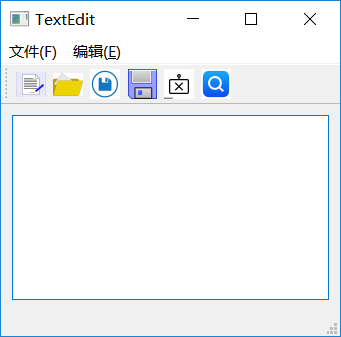
## 作业要求：

基于QMainWindow类和QTextEdit类实现一个文本编辑器，要求至少包含以下功能：

1. 新建、打开、关闭、保存、另存为、查找
2. 菜单栏和工具栏都要有相应图标

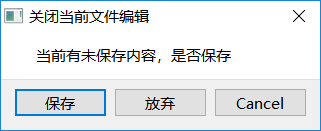
## 使用说明：

1. **主界面：**



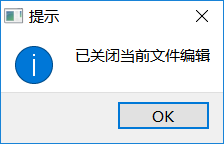
运行文件后会直接进入主界面，此时文档中无内容且未与txt文件绑定。

1. **关闭文件：**
   * 如果此时文档未与txt文件绑定且文档中已经有编辑内容，则跳出如下提示框：

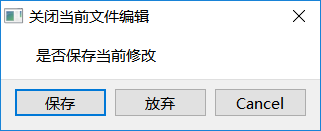


如果用户点保存则进行一次保存操作，如果用户点放弃则将编辑框清空。

* + 如果此时文档未与txt文件绑定，且文档中没有编辑内容，则直接跳出如下提示框：



* + 如果此时文档已经与txt文件绑定，则直接提醒用户是否要保存当前的修改：

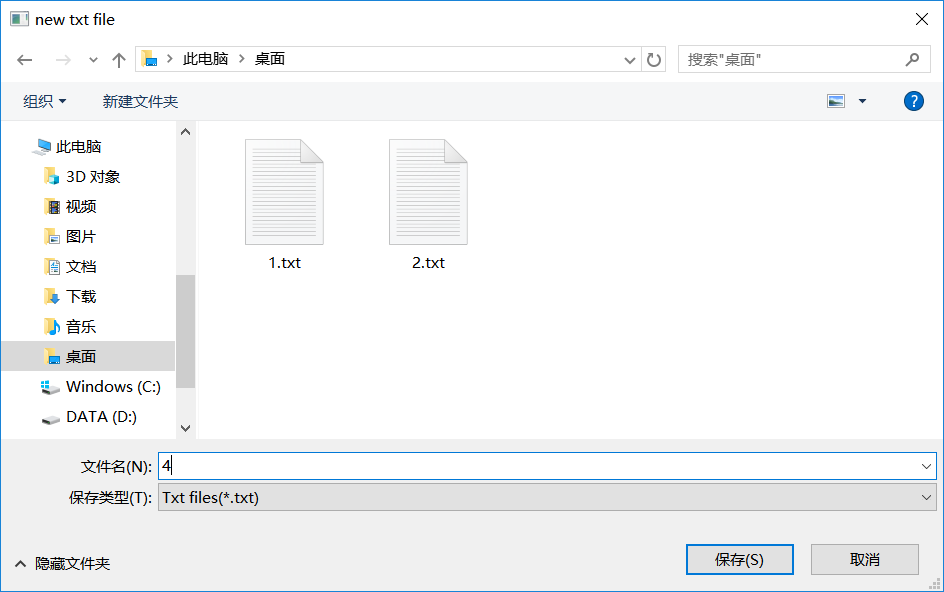


(如果用户编辑框中内容与txt文件中内容没有差异同样会弹出本对话框，起到提醒作用)

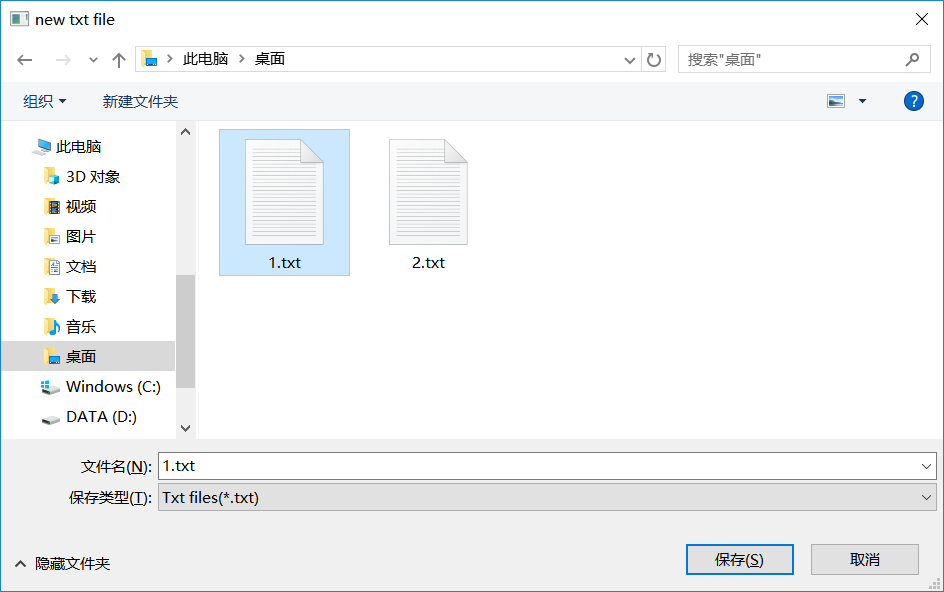
1. **新建文件：**

如果此时文档中有内容或文档已与txt文件绑定，则先执行一边关闭文件操作，再进行新建文件。

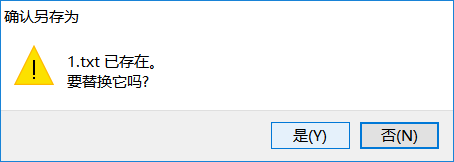
用户直接在弹出的对话框中选定位置，并创建文件：



如果创建的文件名已经存在：



则跳出以下对话框：



如果用户点击是，则将该txt文档置为空。

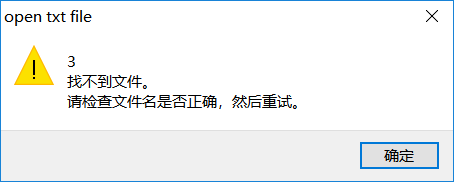
选好文件后，该txt文件将与编辑框绑定。

1. **打开文件：**

如果此时文档中有内容或文档已与txt文件绑定，则先执行一遍关闭文件操作，再进行打开文件。

将选择的文件内容输入至编辑框中，并将编辑框与文件绑定。

如果选择的文件不存在则弹出提示对话框：

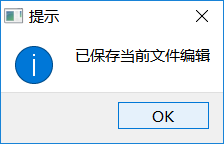


1. **保存文件：**

如果此时文档已经与txt文件绑定，则直接将文件内容保存至该txt文件中。

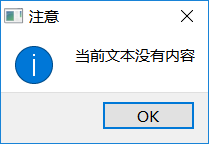
如果此时文档未与txt文件绑定，则会弹出对话框让用户新建一个txt文件，并将该txt文件与文档绑定，随后将文件内容保存至该txt文件中。

如果该操作成功进行，则将弹出提示对话框：



1. **另存为：**

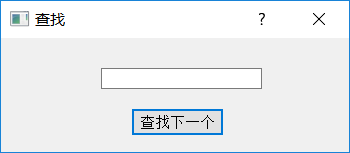
如果此时文档内容为空，则会弹出提示框：



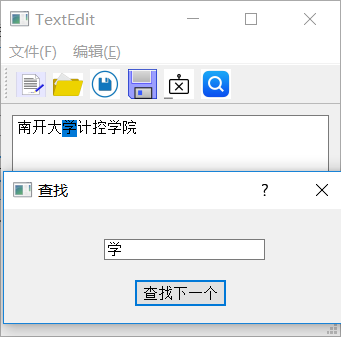
正常情况将会将文档中的内容保存至一个用户指定的新txt文件或者覆盖一个已有的txt文件。

1. **搜索：**

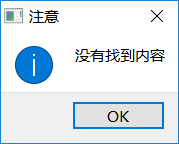
点击搜索后会弹出一个非模态对话框：



用户在该lineEdit中所写入的内容，将在主窗口的文档中进行搜索，搜索到的内容将被标出：



查找将从光标所在位置向后进行搜索，如果其后没有找到所要搜索的内容，则会跳出提示框：



跳出提示框后，光标将会被置于文档的开始位置。

## 实现思路：

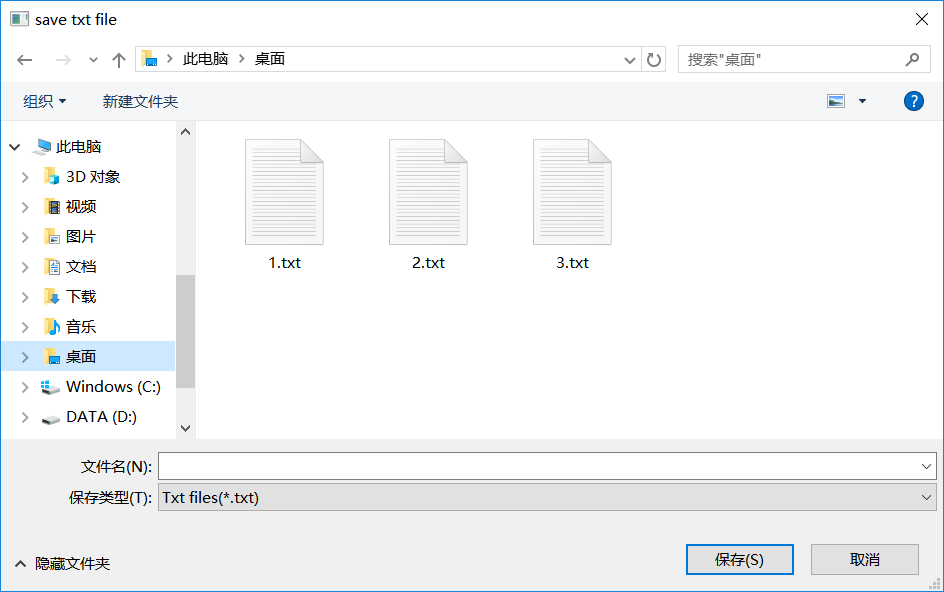
1. **文件保存实现方法：**

[**QString**](../qtcore/qstring.html) QFileDialog::**getSaveFileName**(**[QWidget](qwidget.html" \l "QWidget)** \**parent* = nullptr, const [**QString**](../qtcore/qstring.html) &*caption* = QString(), const [**QString**](../qtcore/qstring.html) &*dir* = QString(), const [**QString**](../qtcore/qstring.html) &*filter* = QString(), [**QString**](../qtcore/qstring.html) \**selectedFilter* = nullptr, [**QFileDialog::Options**](qfiledialog.html#Option-enum) *options* = ...)

***参数值含义：***

* 第一个参数parent，用于指定父组件。注意，很多Qt组件的构造函数都会有这么一个parent参数，并提供一个默认值0；
* 第二个参数caption，是对话框的标题；
* 第三个参数dir，是对话框显示时默认打开的目录，"." 代表程序运行目录，"/" 代表当前盘符的根目录(Windows，Linux下/就是根目录了)，也可以是平台相关的，比如"C:\\"等；
* 第四个参数filter，是对话框的后缀名过滤器，比如我们使用"Image Files(\*.jpg \*.png)"就让它只能显示后缀名是jpg或者png的文件。如果需要使用多个过滤器，使用";;"分割，比如"JPEG Files(\*.jpg);;PNG Files(\*.png)"；
* 第五个参数selectedFilter，是默认选择的过滤器；
* 第六个参数options，是对话框的一些参数设定，比如只显示文件夹等等，它的取值是enum QFileDialog::Option，每个选项可以使用 | 运算组合起来。

调用QFileDialog中的该函数，可以实现打开一个文件保存对话框：



利用该函数返回的文件路径名，创建一个QFile类变量，已只写形式打开：

QFile f1(fileName);

f1.*open*(QIODevice::WriteOnly|QIODevice::Text);

用该变量创建QTextStream流类变量：

QTextStream str(&f1);

str<<ui->textEdit->toPlainText();

将文档中内容写入文件，最后将文件关闭：

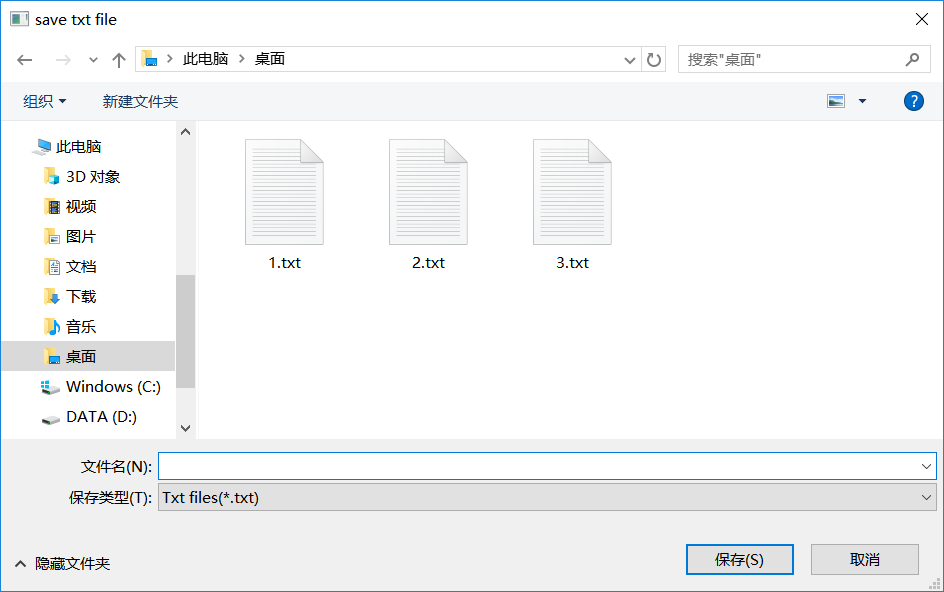
f1.*close*();

1. **文件打开实现方法：**

[**QString**](../qtcore/qstring.html) QFileDialog::**getOpenFileName**(**[QWidget](qwidget.html" \l "QWidget)** \**parent* = nullptr, const [**QString**](../qtcore/qstring.html) &*caption* = QString(), const [**QString**](../qtcore/qstring.html) &*dir* = QString(), const [**QString**](../qtcore/qstring.html) &*filter* = QString(), [**QString**](../qtcore/qstring.html) \**selectedFilter* = nullptr, [**QFileDialog::Options**](qfiledialog.html#Option-enum) *options* = ...)

参数值含义同上

调用QFileDialog中的该函数，可以实现打开一个文件保存对话框：



利用该函数返回的文件路径名，创建一个QFile类变量，已只读形式打开：

QFile file(fileName);

file.*open*(QIODevice::ReadOnly | QIODevice::Text);

将txt文件中的内容逐行读出：

while(!file.*atEnd*())

{

QTextStream txtInput(&file);

QString str=txtInput.readLine();

displayString.append(str);

}

显示在屏幕上：

ui->textEdit->clear();

ui->textEdit->setPlainText(displayString);

1. **弹出对话框：**

除了搜索对话框，本程序中所有弹出的对话框均为调用Qt中QMessagebox类来实现。

* 询问对话框：

**int** QMessageBox::**question**(**[QWidget](qwidget.html" \l "QWidget)** \**parent*, const [**QString**](../qtcore/qstring.html) &*title*, const [**QString**](../qtcore/qstring.html) &*text*, **int** *button0*, **int** *button1* = 0, **int** *button2* = 0)

使用例子：

switch(QMessageBox::question(this,tr("询问"),tr("关闭当前编辑？"),

QMessageBox::Ok|QMessageBox::Cancel,QMessageBox::Ok))

{

case QMessageBox::Ok:

this->on\_action\_C\_triggered();//关闭当前编辑

break;

case QMessageBox::Cancel:

return; //放弃打开新文件

break;

default:

return;

}

}

* 提示对话框：

**int** QMessageBox::**information**(**[QWidget](qwidget.html" \l "QWidget)** \**parent*, const [**QString**](../qtcore/qstring.html) &*title*, const [**QString**](../qtcore/qstring.html) &*text*, **int** *button0*, **int** *button1* = 0, **int** *button2* = 0)

使用例子：

QMessageBox::information(this,tr("提示"),tr("已关闭当前文件编辑"));

* 自定义对话框：

QMessageBox::**QMessageBox**(**[QWidget](qwidget.html" \l "QWidget)** \**parent* = nullptr)

**void** QMessageBox::**addButton**(**[QAbstractButton](qabstractbutton.html)** \**button*, **QMessageBox::ButtonRole** *role*)

**void** QMessageBox::**setWindowTitle**(const [**QString**](../qtcore/qstring.html) &*title*)

使用例子：

QMessageBox MsgBox;

MsgBox.setWindowTitle(tr("关闭当前文件编辑"));

QPushButton \*saveButton = MsgBox.addButton(tr("保存"),QMessageBox::ActionRole);

QPushButton \*unsaveButton = MsgBox.addButton(tr("放弃"),QMessageBox::ActionRole);

QPushButton \*cancelButton = MsgBox.addButton(QMessageBox::Cancel);

1. **文档搜索：**

通过如下函数实现查找到文档中相应内容

**bool** QTextEdit::**find**(const [**QRegExp**](../qtcore/qregexp.html) &*exp*, [**QTextDocument::FindFlags**](../qtgui/qtextdocument.html#FindFlag-enum) *options* = QTextDocument::FindFlags())

通过platte类将相应内容进行标注操作：

QPalette palette = ui->textEdit->palette();

palette.setColor(QPalette::Highlight,palette.color(QPalette::Active,QPalette::Highlight));

ui->textEdit->setPalette(palette);

如果没有查找到相应内容，通过：

**void** QTextEdit::**moveCursor**(**QTextCursor::MoveOperation** *operation*, **QTextCursor::MoveMode** *mode* = QTextCursor::MoveAnchor)

将鼠标移至文档开始位置。

使用例子：

ui->textEdit->moveCursor(QTextCursor::Start, QTextCursor::MoveAnchor);

## 走过的弯路：

在进行txt文件的打开与保存过程中，一开始由于txt文件的编码问题，错误的以二进制文件的形式进行了保存，这种方式如果只在本程序中对文档进行操作，则不会发现问题，而当实际打开保存的txt文件时则会发现，保存的全是乱码。

这是由于txt文件为文本文件，解码方式为ASCII码，所以以二进制形式存储时将会出现乱码。

以下是二进制文件的保存与读取操作所使用代码：

* 读取：

while(!file.*atEnd*())

{

QByteArray line = file.readLine();

QString str(line);

displayString.append(str);

}

* 保存：

QFile f1(fileName);

f1.*open*(QIODevice::WriteOnly|QIODevice::Text);

QByteArray ba;

ba=ui->textEdit->toPlainText().toLatin1();

f1.write(ba);