【文本编辑器】四、文档排版美化功能（下）

原创 Qt 学习 Qt 学习 2020-02-26

本篇对文本排序和文本字体字号设置进行讲解，以及与这些文本格式对应的组合框显示状态设置。

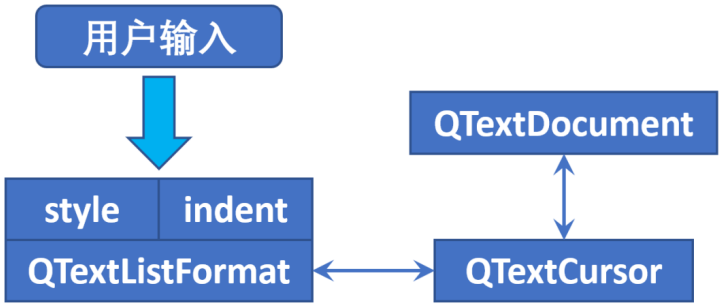
本篇目录

运行环境：

win 10 + Qt 5.12.5 + Qt Creator 4.10

1. 文本排序

文本排序功能的基本流程如图



文本排序功能实现的基本流程

用于描述文本排序格式的 `QTextListFormat` 包含两个基本的属性：

- `QTextListFormat::style`
  表示文本采用哪种排序方式；

- `QTextListFormat::indent`
  表示排序后的缩进值。

因此，若要实现文本排序的功能则只需设置好 `QTextListFormat` 的以上两个属性，并将整个格式通过 `QTextListCursor` 类应用到文本中即可。

在通常的文本编辑器中， `QTextListFormat` 的缩进值 `indent` 都是预设好的，并不需要用户来设定。本实例采用在程序中通过获取当前文本段 `QTextBlockFormat` 的缩进值来进行相应计算的方法，以获得排序文本的缩进值。

在 "textedit.h" 中添加文本排序功能的槽函数

```
1  private slots:
2      void textStyle(int styleIndex);
```

在源文件中添加

```
#include <QTextList>
```

以及根据用户选择的不同排序方式对文本排序功能的实现

```
void TextEdit::textStyle(int styleIndex)
{
    QTextCursor cursor = textEdit->textCursor(); // 获得文本编辑框的 QTextCursor 对象指针
    QTextListFormat::Style style = QTextListFormat::ListStyleUndefined;

    switch (styleIndex) { // 提供了 8 种文本排序的方式
    case 1:
        style = QTextListFormat::ListDisc;
        break;
    case 2:
        style = QTextListFormat::ListCircle;
        break;
    case 3:
        style = QTextListFormat::ListSquare;
        break;
    case 4:
        style = QTextListFormat::ListDecimal;
        break;
    case 5:
        style = QTextListFormat::ListLowerAlpha;
        break;
    case 6:
        style = QTextListFormat::ListUpperAlpha;
        break;
    case 7:
        style = QTextListFormat::ListLowerRoman;
        break;
    case 8:
        style = QTextListFormat::ListUpperRoman;
```

```
            break;
        default:
            break;
        }

        cursor.beginEditBlock(); // 设置缩进值

        QTextBlockFormat blockFmt = cursor.blockFormat();

        if (style == QTextListFormat::ListStyleUndefined) {
            blockFmt.setObjectIndex(-1);
            int headingLevel = styleIndex >= 9 ? styleIndex - 9 + 1 : 0; // H1 to H6, or Standard
            blockFmt.setHeadingLevel(headingLevel);
            cursor.setBlockFormat(blockFmt);

            int sizeAdjustment = headingLevel ? 4 - headingLevel : 0; // H1 to H6: +3 to -2
            QTextCharFormat fmt;
            fmt.setFontWeight(headingLevel ? QFont::Bold : QFont::Normal);
            fmt.setProperty(QTextFormat::FontSizeAdjustment, sizeAdjustment);
            cursor.select(QTextCursor::LineUnderCursor);
            cursor.mergeCharFormat(fmt);
            textEdit->mergeCurrentCharFormat(fmt);
        } else {
            QTextListFormat listFmt;
            if (cursor.currentList()) {
                listFmt = cursor.currentList()->format();
            } else {
                listFmt.setIndent(blockFmt.indent() + 1); // 在段落缩进的基础上加 1
                blockFmt.setIndent(0);
                cursor.setBlockFormat(blockFmt);
            }
            listFmt.setStyle(style);
            cursor.createList(listFmt);
        }

        cursor.endEditBlock();
    }
```

其中，

- `cursor.beginEditBlock();`

  ...

  `cursor.endEditBlock();`

  此代码段完成 `QTextListFormat` 的 `indent` 属性（即缩进值）的设定，并将设置的格式应用到光标所在的文本处。

以 `cursor.beginEditBlock()` 开始，以 `cursor.endEditBlock()` 结束，这两个函数的作用是设定这两个函数之间的所有操作，相当于一个动作，如果需要进行撤销或恢复，则这两个函数直接按的所有操作将同时被撤销或恢复，这两个函数通常成对出现。

设置 `QTextListFormat` 的缩进值首先通过 `QTextCursor` 获得 `QTextBlockFormat` 对象，由 `QTextBlockFormat` 获得段落的缩进值，在此基础上定义 `QTextListFormat` 的缩进值，本实例是在段落缩进的基础上加 1，也可以根据需要进行其它设定。

为了使应用程序能够支持从组合工具栏中选择段落标号和编号类型，在 `setupTextActions()` 函数中将一下代码段中加黑语句前的注释符 "//" 去掉

```
1  void TextEdit::setupTextActions()
2  {
3      // 格式主菜单动作集
4      ...
5
6      comboStyle = new QComboBox(comboToolbar);
7      comboToolbar->addWidget(comboStyle); // 将组合框添加至工具条
8      comboStyle->addItem(tr("标准"));
9      comboStyle->addItem(tr("项目符号 (●)"));
10     comboStyle->addItem(tr("项目符号 (○)"));
11     comboStyle->addItem(tr("项目符号 (■)"));
12     comboStyle->addItem(tr("项目编号 (1.2.3.)"));
13     comboStyle->addItem(tr("项目编号 (a.b.c.)"));
14     comboStyle->addItem(tr("项目编号 (A.B.C.)"));
15     comboStyle->addItem(tr("项目编号 (i.ii.iii.)"));
```

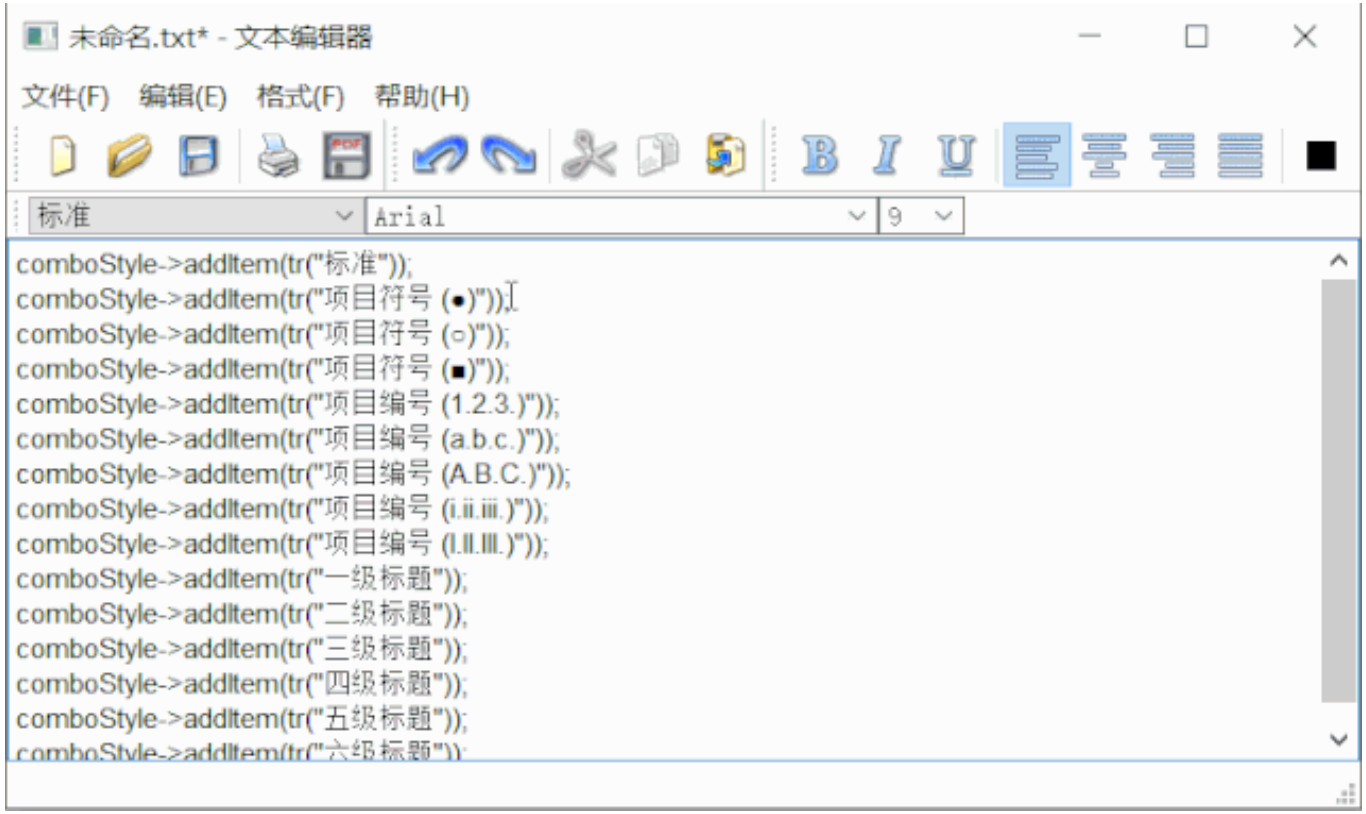
```
    comboStyle->addItem(tr("项目编号 (I.II.III.)"));
    comboStyle->addItem(tr("一级标题"));
    comboStyle->addItem(tr("二级标题"));
    comboStyle->addItem(tr("三级标题"));
    comboStyle->addItem(tr("四级标题"));
    comboStyle->addItem(tr("五级标题"));
    comboStyle->addItem(tr("六级标题"));
    comboStyle->setStatusTip(tr("添加项目符号（编号）或设置段落等级"));
    connect(comboStyle, QOverload<int>::of(&QComboBox::activated), this, &TextEdit::textStyle);

    ...
}
```

运行程序，可以实现文本段落标号和编号。接下来还需设置组合框的显示状态，即将当前光标处的文本格式在组合框中显示出来。按照下面的代码将 `cursorPositionChanged()` 函数补充完善

```
void TextEdit::cursorPositionChanged()
{
    alignmentChanged(textEdit->alignment());
    QTextList *list = textEdit->textCursor().currentList();
    if (list) {
        switch (list->format().style()) {
        case QTextListFormat::ListDisc:
            comboStyle->setCurrentIndex(1);
            break;
        case QTextListFormat::ListCircle:
            comboStyle->setCurrentIndex(2);
            break;
        case QTextListFormat::ListSquare:
            comboStyle->setCurrentIndex(3);
            break;
        case QTextListFormat::ListDecimal:
            comboStyle->setCurrentIndex(4);
            break;
        case QTextListFormat::ListLowerAlpha:
            comboStyle->setCurrentIndex(5);
```

```
            break;
        case QTextListFormat::ListUpperAlpha:
            comboStyle->setCurrentIndex(6);
            break;
        case QTextListFormat::ListLowerRoman:
            comboStyle->setCurrentIndex(7);
            break;
        case QTextListFormat::ListUpperRoman:
            comboStyle->setCurrentIndex(8);
            break;
        default:
            comboStyle->setCurrentIndex(-1);
            break;
        }
    } else {
        int headingLevel = textEdit->textCursor().blockFormat().headingLevel();
        comboStyle->setCurrentIndex(headingLevel ? headingLevel + 8 : 0);
    }
}
```

运行程序，文本排序效果如下



2. 文本字体和字号

在 "textedit.h" 文件中添加设置文本字体和字号的槽函数

```
1  private slots:
2      void textFamily(const QString &f);
3      void textSize(const QString &p);
```

完成选定文本字体和字号设置函数的实现代码如下

```
void TextEdit::textFamily(const QString &f)
{
    QTextCharFormat fmt;
    fmt.setFontFamily(f); // 设置为选择的字体
    mergeFormatOnWordOrSelection(fmt); // 将新的格式应用到光标选区的文本
}

void TextEdit::textSize(const QString &p)
{
    qreal pointSize = p.toDouble();
    if (p.toFloat() > 0) {
        QTextCharFormat fmt;
        fmt.setFontPointSize(pointSize); // 设置为选择的字号
        mergeFormatOnWordOrSelection(fmt);
    }
}
```

添加设置文本字体和字号的 `connect` 函数

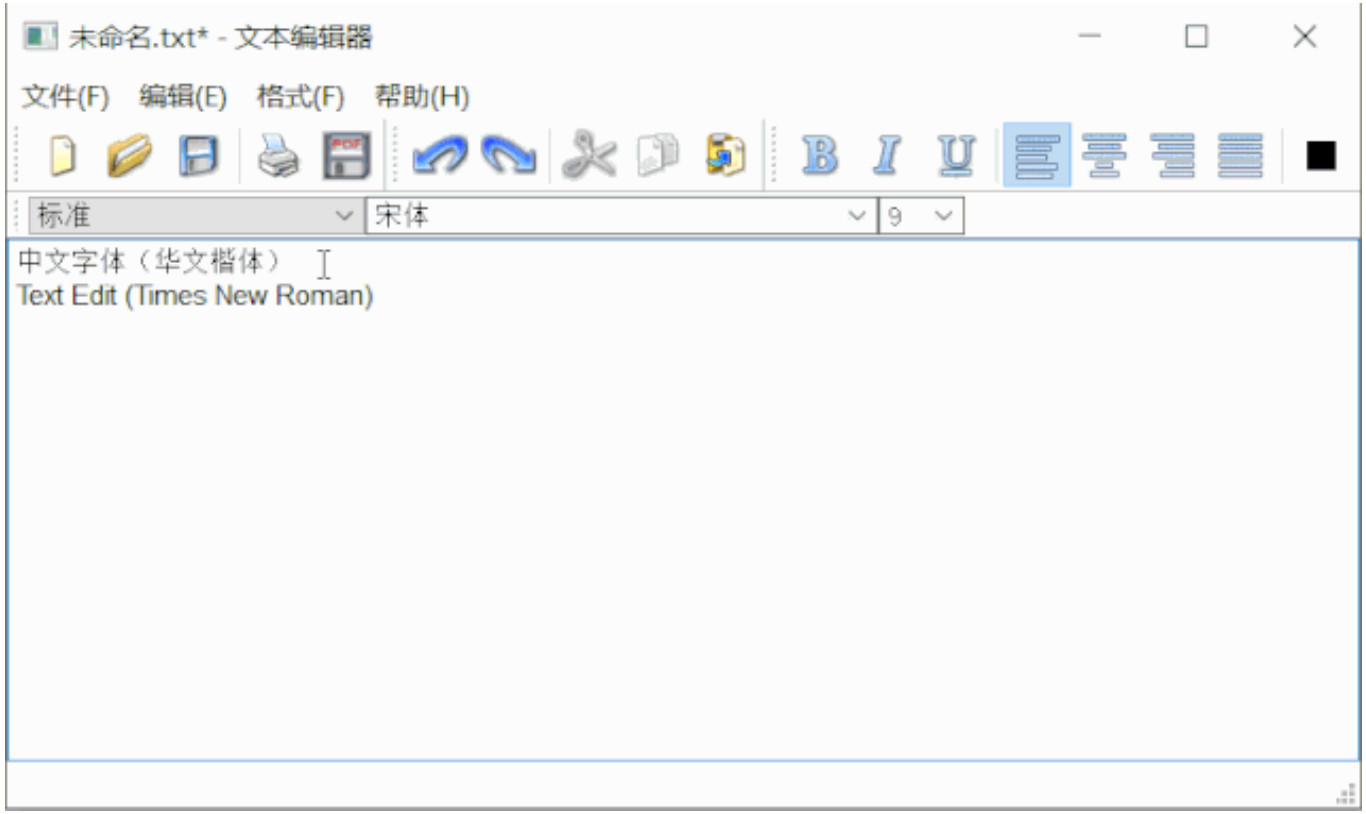
```
void TextEdit::setupTextActions()
{
    // 格式主菜单动作集
    ...

    comboFont = new QFontComboBox(comboToolbar);
    comboToolbar->addWidget(comboFont);  // 将组合框添加至工具条
    comboFont->setStatusTip(tr("更改字体"));
    connect(comboFont, QOverload<const QString &>::of(&QComboBox::activated), this, &TextEdit::textFamily);

    comboSize = new QComboBox(comboToolbar);
    comboToolbar->addWidget(comboSize); // 将组合框添加至工具条
    comboSize->setStatusTip(tr("更改字号"));
    comboSize->setEditable(true);       // 将字号组合框设置为可编辑

    const QList<int> standardSizes = QFontDatabase::standardSizes();
    foreach (int size, standardSizes)
        comboSize->addItem(QString::number(size));
```

```
19      comboSize->setCurrentIndex(standardSizes.indexOf(QApplication::font().pointSize()));
20
21      connect(comboSize, QOverload<const QString &>::of(&QComboBox::activated), this, &TextEdit::textSize);
22  }
```

在 `fontChanged()` 函数中设置字体和字号组合框显示格式状态（加粗行代码）

```
1   void TextEdit::fontChanged(const QFont &f)
2   {
3       comboFont->setCurrentIndex(comboFont->findText(QFontInfo(f).family()));
4       comboSize->setCurrentIndex(comboSize->findText(QString::number(f.pointSize())));
5       actionTextBold->setChecked(f.bold());
6       actionTextItalic->setChecked(f.italic());
7       actionTextUnderline->setChecked(f.underline());
8   }
```

运行程序，文本字体和字号设置效果如下



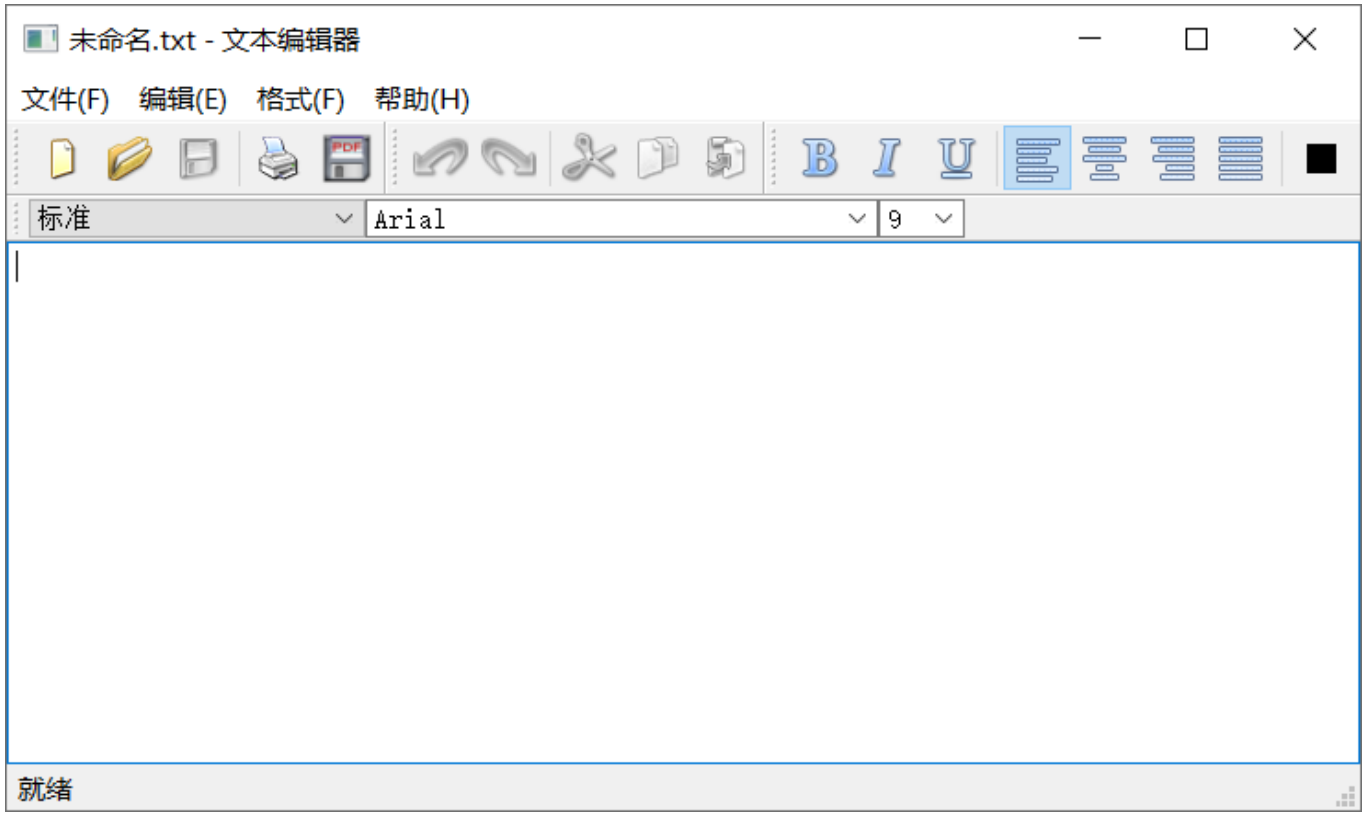
3. 设置初始字体

在构造函数中设置文本编辑框中的初始字体

```
TextEdit::TextEdit(QWidget *parent)
    : QMainWindow(parent)
{
    ...

    QFont textFont("Helvetica");
    textFont.setStyleHint(QFont::SansSerif);
    textEdit->setFont(textFont);
```

```
 9       fontChanged(textEdit->font());
10       colorChanged(textEdit->textColor());
11       alignmentChanged(textEdit->alignment());
12
13       ...
14   }
```

运行程序，初始界面中的组合框显示如下



小结

本篇实现了文本排序和文本字体字号设置操作，以及与这些文本格式对应的组合框显示状态设置。

相关阅读：

《【文本编辑器】一、界面设计》
《【文本编辑器】二、文件操作功能（上）》
《【文本编辑器】二、文件操作功能（下）》
《【文本编辑器】三、文本编辑功能》
《【文本编辑器】四、文档排版美化功能（上）》