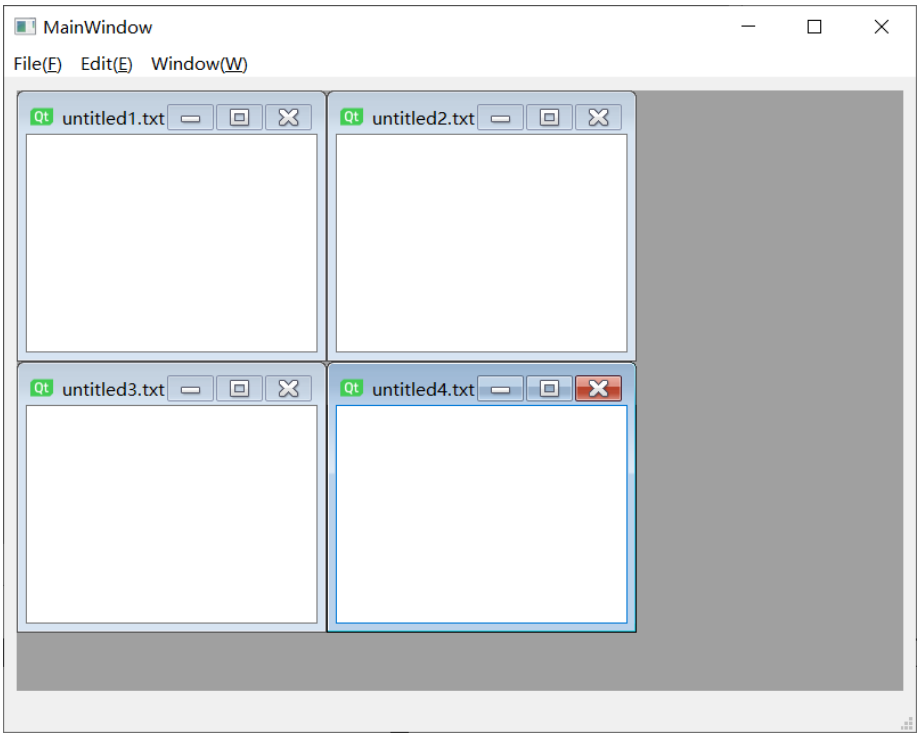
第三次课程设计——文档编辑器

一、主要内容

此次课程设计较规范地实现了一个多文档编辑器

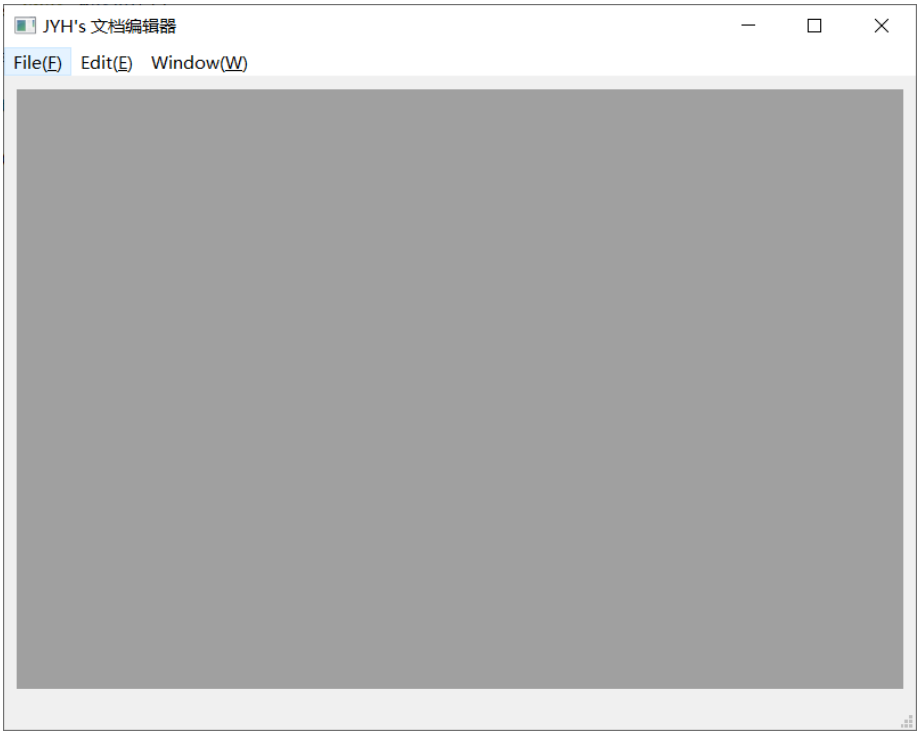


类似于记事本，可以实时输入文本。支持txt文件的读取、保存、撤销等工作等。

二、设计思路

1.界面设计

先进行主窗口菜单栏的设计。

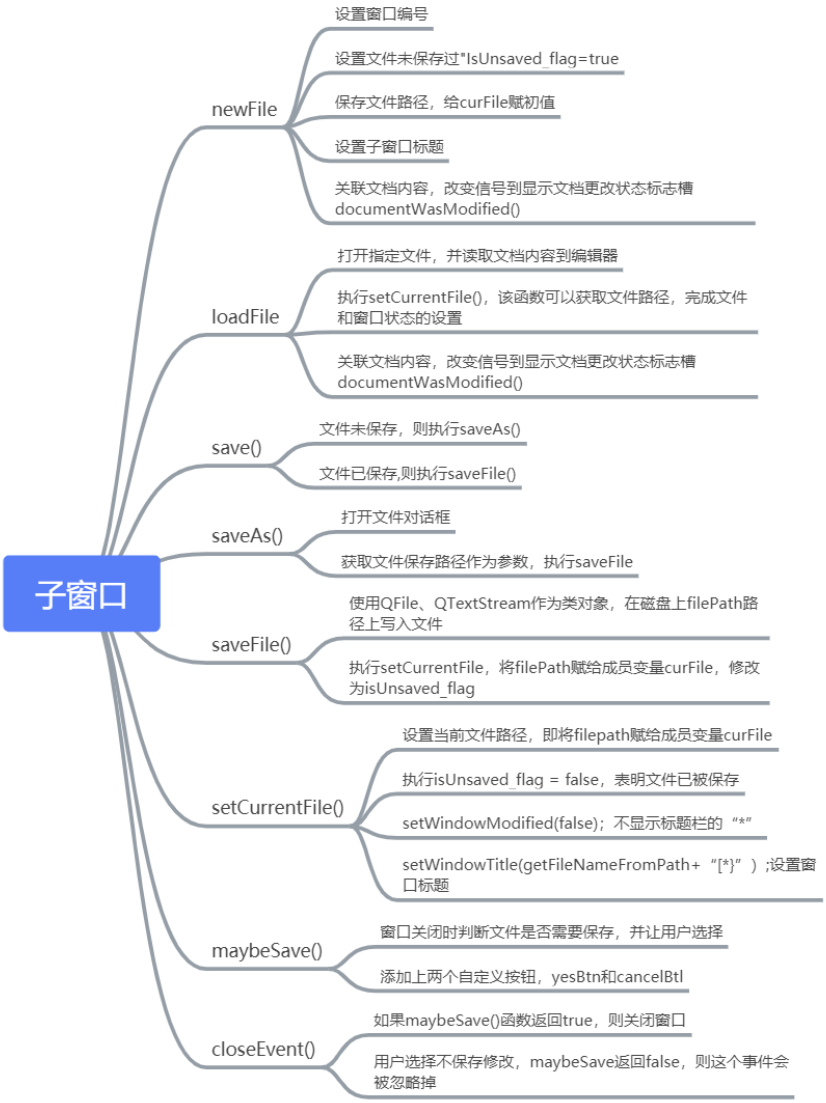


设计完菜单栏，向Action Editor添加子菜单。

2.子窗口设计

子窗口中心部件使用`QTextEdit`类,故实现一个继承于`QTextEdit`的类，类函数由思维导图提供

如下：



3.实现菜单的功能

1.更新菜单的状态

通过窗口的状态确定，哪些功能键可以使用，哪些不能。

```
#include <QMainWindow>

//增加类MdiChild的前置声明
class MdiChild;

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

```

```
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_actionNew_triggered();
    void updateMenus();            //更新菜单

private:
    Ui::MainWindow *ui;
    QAction *actionSeparator;    //分隔符
    MdiChild *activeMdiChild(); //活动窗口
};
```

2.实现新建文件的操作

```
MdiChild * MainWindow::createMdiChild() //创建子窗口部件
{
    MdiChild *child = new MdiChild; //创建MdiChild部件
    ui->mdiArea->addSubWindow(child); //向多文档区域添加子窗口，child为中心部件
    connect(child,SIGNAL(copyAvailable(bool)),ui->actionCut,
            SLOT(setEnabled(bool)));

    // 根据QTextEdit类的是否可以复制信号设置剪切复制动作是否可用
    connect(child,SIGNAL(copyAvailable(bool)),ui->actionCopy,
            SLOT(setEnabled(bool)));

    // 根据QTextDocument类的是否可以撤销恢复信号设置撤销恢复动作是否可用
    connect(child->document(),SIGNAL(undoAvailable(bool)),
            ui->actionUndo,SLOT(setEnabled(bool)));
    connect(child->document(),SIGNAL(redoAvailable(bool)),
            ui->actionRedo,SLOT(setEnabled(bool)));

    return child;
}
```

3.实现文件打开操作

```
void MainWindow::on_actionOpen_triggered() // 打开文件菜单
{
    QString filePath = QFileDialog::getOpenFileName(this); // 获取文件路径
```

```
    if (!filePath.isEmpty())
    {
        // 如果路径不为空，则查看该文件是否已经打开
        QMdiSubWindow *existing = findMdiChild(filePath);
        if (existing)
        {
            // 如果已经存在，则将对应的子窗口设置为活动窗口
            ui->mdiArea->setActiveSubWindow(existing);
            return;
        }

        MdiChild *child = createMdiChild(); // 如果没有打开，则新建子窗口
        if (child->loadFile(filePath))
        {
            ui->statusBar->showMessage(tr("打开文件成功"), 2000);
            child->show();
        }
        else
        {
            child->close();
        }
    }
}

```

4.添加子窗口列表

```
void MainWindow::updateWindowMenu() // 更新窗口菜单
{
    ui->menuW->clear(); // 先清空菜单，然后再添加各个菜单动作
    ui->menuW->addAction(ui->actionClose);
    ui->menuW->addAction(ui->actionCloseAll);
    ui->menuW->addSeparator();
    ui->menuW->addAction(ui->actionTile);
    ui->menuW->addAction(ui->actionCascade);
    ui->menuW->addSeparator();
    ui->menuW->addAction(ui->actionNext);
    ui->menuW->addAction(ui->actionPrevious);
    ui->menuW->addAction(actionSeparator);

    QList<QMdiSubWindow *> windows = ui->mdiArea->subWindowList();
    actionSeparator->setVisible(!windows.isEmpty());
    // 如果有活动窗口，则显示间隔器
    for (int i = 0; i < windows.size(); ++i)
    {
```

```
19          // 遍历各个子窗口
20          MdiChild *child = qobject_cast<MdiChild *>
(windows.at(i)->widget());
21
22          QString text;
23          if (i < 9)
24          {
25              // 如果窗口数小于9，则设置编号为快捷键
26              text = tr("&%1 %2").arg(i + 1)
27                                 .arg(child-
>getFileNameFromPath());
28          }
29          else
30          {
31              text = tr("%1 %2").arg(i + 1)
32                                .arg(child-
>getFileNameFromPath());
33          }
34          QAction *action  = ui->menuW-
>addAction(text); // 添加动作到菜单
35          action->setCheckable(true); // 设置动作可以选择
36
37          // 设置当前活动窗口动作为选中状态
38          action ->setChecked(child ==
activeMdiChild());
39
40          // 关联动作的触发信号到信号映射器的map()槽函数上，这
个函数会发射mapped()信号
41          connect(action, SIGNAL(triggered()),
windowMapper, SLOT(map()));
42
43          // 将动作与相应的窗口部件进行映射，在发射mapped()信号
时就会以这个窗口部件为参数
44          windowMapper->setMapping(action,
windows.at(i));
45
46      }
47  }
```

5.其它功能

因为在前面已经把核心的功能都实现了，而且像剪切、复制、撤销等常用功能，`QTextEdit`类已经提供了，所以这里只需要调用相应的函数即可。

三、遇到的问题与解决方案

1.文件名问题

实现文档编辑器，需要在未保存的文件名上加上(*),

这里使用了一个`document()`的库函数

根据文档的`isModified()`函数的返回值，判断我们编辑器内容是否被更改了。如果被更改了，参数为true，则`setWindowModified()`就会在设置了[*]号的地方显示“*”号 `setWindowModified(document()->isModified());`
`setWindowModified()`为库函数

2.添加子窗口列表

我们想每添加一个子窗口就可以在窗口菜单中罗列出它的文件名，而且可以在这个列表中选择一个子窗口，将它设置为活动窗口。这个看似很好实现，只要为窗口菜单添加菜单动作，然后关联这个动作的触发信号到设置活动窗口槽上就可以了。但是，如果有很多个子窗口怎么办，难道要一个一个进行关联吗，那怎么获知是哪个动作？这里使用了一个信号映射器`QSignalMapper`类，它可以实现对多个相同部件的相同信号进行映射，为其添加字符串或者数值参数，然后再发射出去。

这里遍历了多文档区域的各个子窗口，然后以它们中的文件名为文本创建了动作，并将这些动作添加到窗口菜单中。我们将动作的触发信号关联到信号映射器的`map()`槽上，然后设置了动作与其对应的子窗口之间的映射，这样触发菜单时就会执行`map()`函数，而它又会发射`mapped()`信号，这个`mapped()`函数会以子窗口部件为参数，因为在构造函数中设置了这个信号与`setActiveSubWindow()`函数的关联，所以最终会执行设置活动子窗口函数，并且设置选择的动作指定的子窗口为活动窗口