# 面向对象程序设计第一次作业

## ——功能描述与需求分析

宋宇轩 2019K8009929042

**写在前面：**

这是第一次作业的报告文档，主要介绍了我想要实现的项目的基本功能与实现方法。由于我先完成了对类的设计，之后才查阅资料并最终决定利用 QT 库辅助设计，因此在项目实现过程中的部分细节可能与当前的想法不完全相同，具体差异我会在后续的设计文档中体现出来。另外，我觉得对工具的掌握也算是项目实现方法的一部分，因此我将对 QT 的学习与分析也放在了这次报告之中。

## 一. 需求分析与进度安排

我想要实现的程序是一个日记软件，支持单篇日记的文字排版、存储和记录日期等功能，同时可以对多篇日记进行统一查看和检索。预想的界面如下图所示：

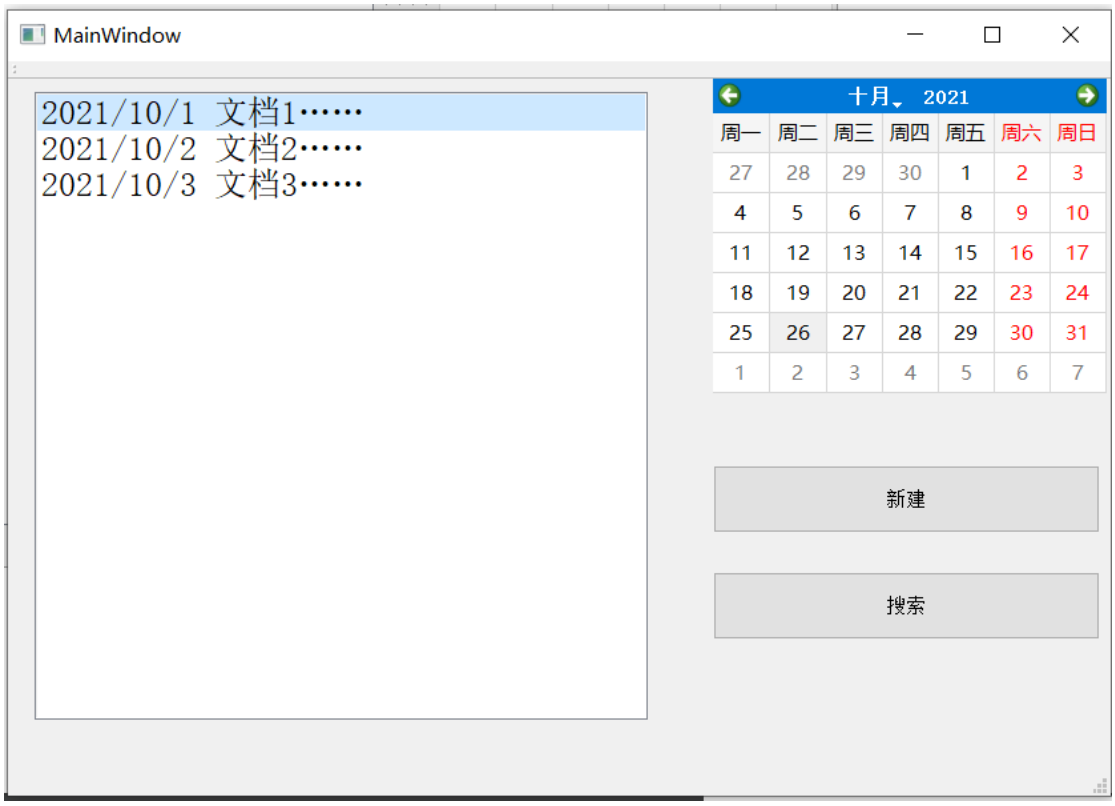


图 1 初始界面示意图

主界面主要可以分为 3 个部分：左侧按时间先后顺序显示目前已有的所有文档的简要信息，单击后即可打开对应文档。右上角是一个日历，右下角则为新建文档和按关键字搜索文档功能的按键。在单击日历中的某一天后，左侧的显示区会显示生成日期为当天的所有文

档，如下图所示：

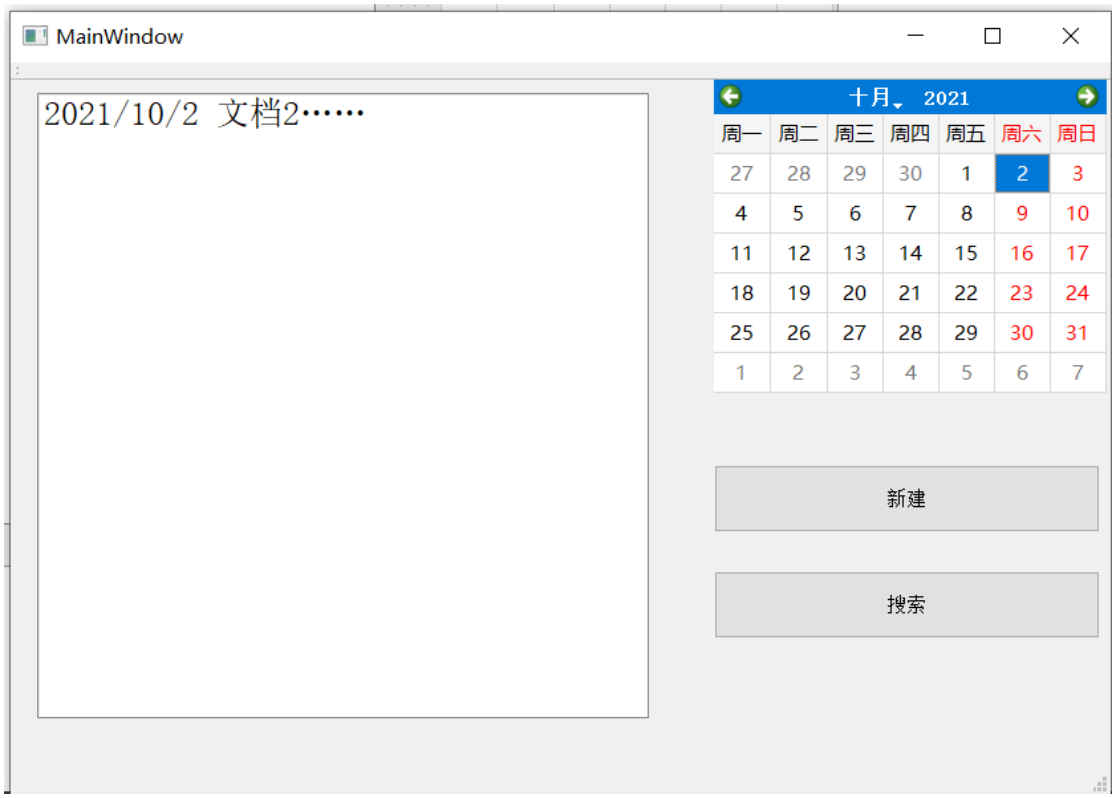


图 2 按日期检索文档示意图

新建文档或打开文档后的界面与 windows 的记事本应用界面类似，支持中英文（UTF-8 编码字符）的编辑、排版、复制粘贴和保存。同时删除文档也将在打开文档之后的子页面内实现。后续将根据时间和进度安排尝试增加实时自动保存，插入图片、视频和 emoji 表情图，支持云端存储，与其它设备互联同步等功能。

目前的进度安排为以三次提交课程作业的时限为节点，将整个学期划分为 3 个阶段。第一阶段确定主要设计目标和需求，借助网上的资料敲定基本目标的实现方法；第二阶段进行程序编写，完成基本功能的大致实现；第三阶段完善基本功能，确保基本功能正确实现，随后尝试加入更多高级功能。

## 二. 设计目标与思路

整个项目可以被分为两个模块：显示总体信息的主界面和提供文本编辑的子界面。主界面的目标状态如第一节所示，提供所有文档的总体展示，支持按日期检索和按关键字搜索，可以新建文档或打开文档。除此之外，还可以在初次打开文档时以只读模式展示，在选择编辑模式之后再启动文档的编辑。子界面将会是一个类似记事本的文本编辑器，用来书写、修改文档，可以支持中英文（UTF-8）的编辑，兼容鼠标操作（通过单击改变光标位置等），支

持手动保存或自动保存，在关闭窗口时提示是否需要保存。除此之外，我还希望能支持撤销功能，实现复制粘贴功能（兼容系统本身的剪切板），为需要的功能增加快捷键（按下 ctrl+s 进行保存等），插入图片、视频或表情，统计字数，修改字体字号，实现文本内容搜索等功能。

目前看来我需要定义三个类：文本类 Text 和目录类 Menu，分别用于对单个文档进行编辑和保存，以及对所有文档进行管理和检索；窗口控制类 Window 用于显示文本，接收键盘、鼠标的操作并调用对应的处理函数。

为了实现文档的编辑，文本类 Text 内需要用来记录文字的缓冲区 text_buff；我计划将每一篇生成的文档都使用.txt 类型文件存储在本地，因此 Text 内需要记录文档所在地址 text_addr；除此之外还需要记录文档的基本信息，例如生成日期 creat_date，总字数 tot_word；为了实现提醒保存，还需要变量 is_change 来记录是否进行了修改。

对文档的操作包含新建，打开，编辑，保存，删除，因此需要 5 个公共的成员函数来实现它们。新建函数 new_text()即是 Text 类的构造函数，负责设置存储地址，初始化生成日期、缓冲区及光标位置等基本信息。打开函数 open_text()根据类中的 text_addr 变量寻址并打开对应文件，将数据写入缓冲区 text_buff 以供编辑。编辑函数 edit_text()需要接收控制部件传来的指令码，并依据指令码对缓冲区做相应的修改（插入字符、删除字符等）。保存函数 save_text()根据原有地址或新输入的地址，将缓冲区中的内容输出到对应地址的文件中。对于删除操作 del_text()，我计划单独设置一个“已删除”文件夹，删除文档时将文档从原地址删除，将其放入这个文件夹之中并修改地址变量，再次删除时才调用析构函数。为此，类中还需要一个成员变量 is_delete 来判断是否已经删除过一次。同时为了支持目录类的管理和检索，还需要输出生成时间和所在地址的函数 get_date()和 get_addr()。

目录类 Menu 将包含一个 Text 类型的优先队列，按照生成时间的先后顺序排序，同时记录目前文档的总数 tot_file。成员函数将包括增加文档（新文档入队），删除文档（从队列中删除对应文档），按时间检索和按文本检索。按时间检索只需要枚举队列里的文档，找到生成时间相符的即可。对于按文本内容检索，我目前的想法是将现有的所有文本依次打开，检查内容中有无检索关键词，之后再关闭文本。但这样需要多次访问内存，对于执行速度可能会有影响，未来可以进一步寻找更优秀的实现方法。

窗口控制类 Window 则包含光标位置 cursor_x 和 cursor_y，以及当前需要执行的指令码 instruction。成员函数有用来接收键盘、鼠标输入并确定目标操作的 check_inst()，根据操作

调用相应处理函数的 handle_inst()，打印当前文档缓冲区的 print_buff()，以及移动光标的 move_cursor()。

综上所述，三个类的 UML 图如下所示：

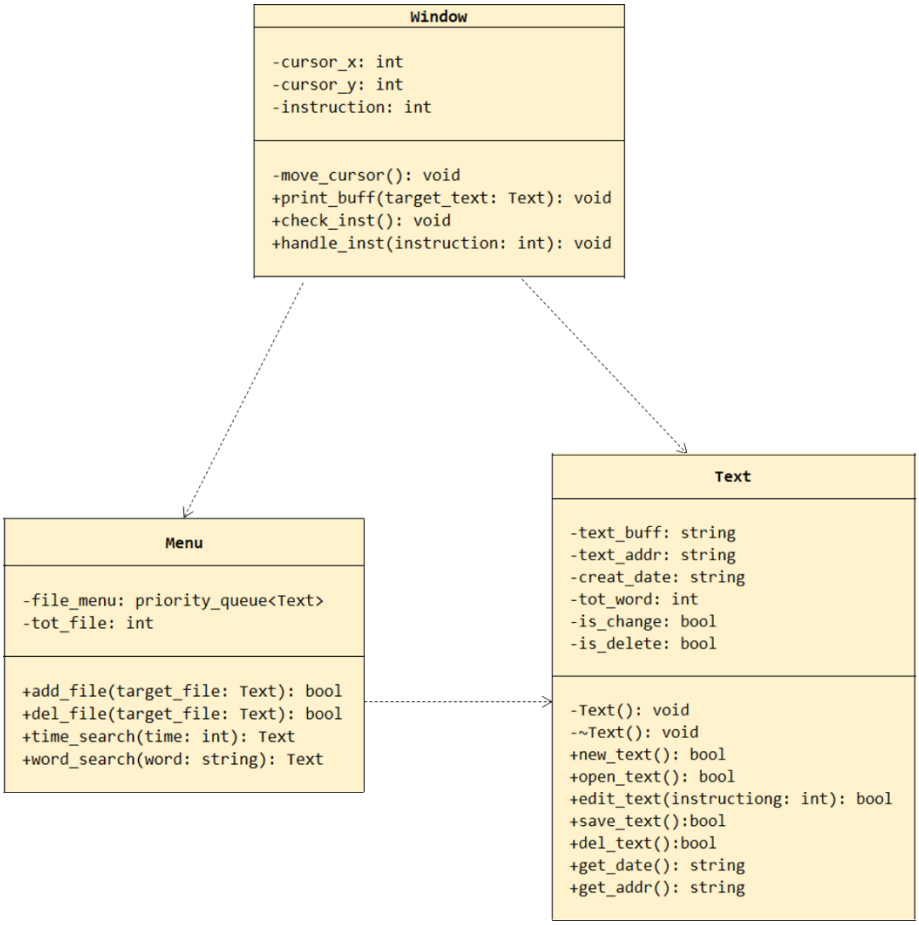


图 3 程序设计 UML 图

## 三． QT 简介与窗口实现分析

在介绍真正的程序设计之前，首先要说明的一点是我使用了 C++语言和 QT 开发库来进行设计。使用 C++语言而非 JAVA 语言主要是因为我对 C++语言相对更为熟悉，不需要重新学习一门新的编程语言，同时设计这样一个规模稍大一些的程序也可以加深我对 C++的理解。使用 QT 库则可以更方便的实现图形化界面，同时对鼠标、键盘操作的支持也更简单，免去了查询 window API 的麻烦。

QT 设计中可以直接拖动功能控件来图形化地实现弹出窗口的 UI 设计，如下图所示：

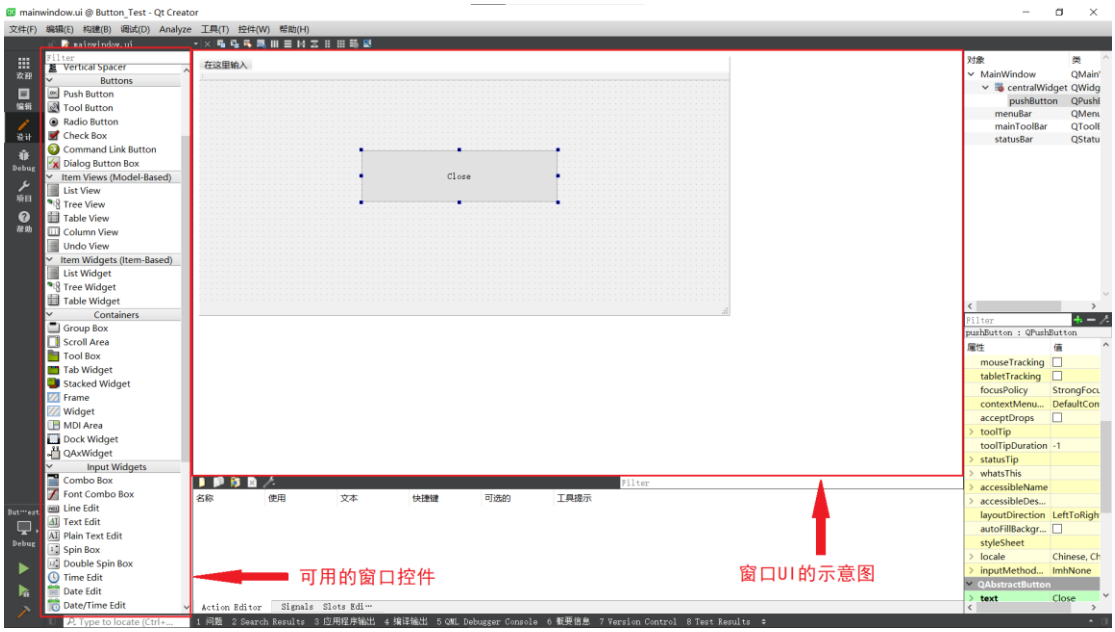


图 4 使用 QT 设计弹出窗口 UI 示意图

QT 软件会根据设计出的界面自动创建一个 UI 类，包含先前加入窗口设计的各类按键、菜单、图表等控件的实现。这些控件都是 QT 已经封装好的类，并提供了需要的接口供设计者使用。以单击窗口中的按钮来关闭窗口为例：从零开始设计需要支持输出图形化界面，检测鼠标指针位置，检测单击动作，设计关闭窗口函数来停止运行窗口等功能，而这些功能大多需要调用 windows API 实现，这是我非常陌生的一块内容；而使用 QT 进行设计只需要在已有的 on_pushButton_clicked()函数中调用 QT 封装好的 close()函数，即可实现单击按钮时关闭窗口的功能，设计者不需要再担心更底层的实现，从而节省时间和精力。事实上，在正式开始编写程序代码之前，我就可以利用 QT 设计生成图 1 和图 2 那样的窗口了，随后只需要对各类控件的功能进行编程实现即可，这也从侧面证明了使用 QT 对于设计效率的巨大提升。

不过尽管利用 QT 进行设计如此便捷，多了解一下窗口和控件的具体实现也是有帮助的。还是以单击窗口中的按钮来关闭窗口为例，这一项目最重要的代码文件有 4 个：main.cpp, ui_mainwindow.h, mainwindow.h 和 mainwindow.cpp。

main.cpp 实现的功能非常简单：定义一个 QApplication 类的对象 a 用于运行程序，一个 MainWindow 类的对象 w 用于显示图形化界面，具体代码如下：

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
```

```
    w.show();

    return a.exec();
}
```

其中 QApplication 的父类为 QGuiApplication，而 QGuiApplication 则继承自基类 QCoreApplication，其主要作用为设置软件图标、接收鼠标和键盘的输入、弹出提示弹窗等较为底层的操作实现。MainWindow 的父类为 QMainWindow，再往上的父类为 QWidget，QWidget 多继承自 QPaintDevice 类和 QObject 类，其中 QPaintDevice 控制弹出窗口的位置、大小、颜色等特征，QObject 则负责将一个信号（比如鼠标的单击）与对应的槽函数（类似处理函数）进行连接，保证对应操作的实现。

信号可以比作软件的中断，例如，按钮控件 A 被点击时就会广播“被单击”这一事件对应的信号。控件中可以有很多信号，每个信号代表不同的事件，而我们只需要向程序指定这一控件对哪些事件感兴趣即可。对于不感兴趣的事件的信号，程序就会忽略掉，不作响应。槽函数则是程序接收并响应某信号之后，需要执行的操作。QT 提供了一些已经设计好了的槽函数，设计者也可以自定义新的槽函数。

ui_mainwindow.h 中声明了 UI_MainWindow 类，包含了对于窗口界面的 UI 设计：

```
QWidget *centralWidget;
QPushButton *pushButton;
QMenuBar *menuBar;
QToolBar *mainToolBar;
QStatusBar *statusBar;
```

其中中心窗口、菜单栏、工具栏和状态栏是每个窗口 UI 都有的，按键控件*pushButton 则是依照我的设计而自动添加的。

此外这个类中还包含两个成员函数：setupUI 和 retranslateUI。retranslateUI 的作用是修改窗口与控件的显示文本（即窗口的标题、按键上显示的文字等）：

```
void retranslateUi(QMainWindow *MainWindow)
{
    MainWindow->setWindowTitle(QCoreApplication::translate("MainWindow",
"MainWindow", nullptr));
    pushButton->setText(QCoreApplication::translate("MainWindow",  "Close",
nullptr));
}
```

setupUI 可以分为三个部分：第一部分是设置窗口与控件的大小、位置、命名等特征，第二部分是调用 retranslateUI 函数设置窗口与控件的显示文本，第三部分是调用 QMetaObject::connectSlotsByName(MainWindow);函数来将当前窗口中的控件与其对应的槽

函数连接起来：

```
void setupUi(QMainWindow *MainWindow)
{
    if (MainWindow->objectName().isEmpty())
        MainWindow->setObjectName(QString::fromUtf8("MainWindow"));
    MainWindow->resize(918, 446);
    centralWidget = new QWidget(MainWindow);
    centralWidget->setObjectName(QString::fromUtf8("centralWidget"));
    pushButton = new QPushButton(centralWidget);
    pushButton->setObjectName(QString::fromUtf8("pushButton"));
    pushButton->setGeometry(QRect(280, 120, 341, 91));
    MainWindow->setCentralWidget(centralWidget);
    menuBar = new QMenuBar(MainWindow);
    menuBar->setObjectName(QString::fromUtf8("menuBar"));
    menuBar->setGeometry(QRect(0, 0, 918, 26));
    MainWindow->setMenuBar(menuBar);
    mainToolBar = new QToolBar(MainWindow);
    mainToolBar->setObjectName(QString::fromUtf8("mainToolBar"));
    MainWindow->addToolBar(Qt::TopToolBarArea, mainToolBar);
    statusBar = new QStatusBar(MainWindow);
    statusBar->setObjectName(QString::fromUtf8("statusBar"));
    MainWindow->setStatusBar(statusBar);

    retranslateUi(MainWindow);

    QMetaObject::connectSlotsByName(MainWindow);
}
```

mainwindow.h 文件中主要声明了表示图形化窗口的 MainWindow 类，其中除了构造函数和析构函数之外，还声明了需要用到的槽函数，并定义了对象*ui 来表示当前窗口：

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_pushButton_clicked();

private:
    Ui::MainWindow *ui;
};
```

mainwindow.cpp 用于定义 MainWindow 类里的成员函数，这也是利用 QT 设计程序时主要需要编写的部分。除了自带的构造函数和析构函数以外，这个程序里增加了按键与一个槽函数 on_pushButton_clicked()，这个函数调用 QT 提供的 close()函数来关闭窗当前口，具体代码如下：

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    close();
}
```

以上就是 QT 在设计好一个窗口后自动生成的主要代码了，接下来需要实现的就是根据需求安排控件和信号，并编写相应的槽函数来实现对应的功能。

## 参考文献


对 C++的简单介绍：https://www.runoob.com/cplusplus/cpp-inheritance.html

从 0 开始设计文本编辑器：https://www.catch22.net/tuts/neatpad#

UML 图箭头的意义：https://blog.csdn.net/wglla/article/details/52225571

对 QT 的简介和初步使用：http://c.biancheng.net/view/1804.html

在线查看 QT 设计源码：https://code.woboq.org/qt5/

利用 QT 设计文本编辑器：https://www.bilibili.com/video/BV1Lo4y1Z7hz?p=7

对信号和槽函数的介绍：https://blog.csdn.net/u014453898/article/details/70242786