# 1 QT写代码的框架

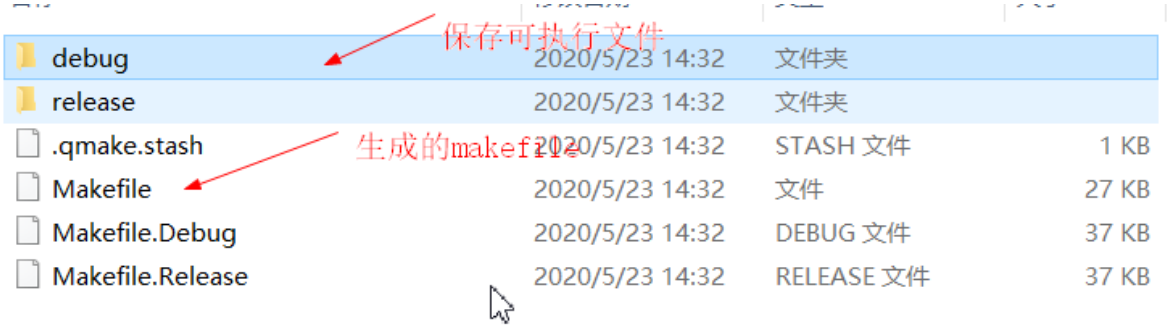
```
#include <QApplication> //QT的框架头文件

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);//QT的框架初始化


  return a.exec();//a.exec()作用是让程序不死 类似于while(1)循环 循环检测事件的产生
  }

```

# 2 编译时产生的文件


# 3 创建空项目

```
#include <QApplication> //QT的框架头文件
#include <QWidget>
```

```
#include <QDebug>
int main(int argc, char *argv[])
{
 QApplication a(argc, argv);//QT的框架初始化
 //QWidget
 QWidget win;
 win.resize(400,300);
 win.setWindowTitle("hello qt");
 win.show();//显示
 qDebug()<<"hello";

 return a.exec();//a.exec()作用是让程序不死 类似于while(1)循环 循环检测事件的产生
 }

```

QT中的命名规则:

对于类的命名 大驼峰法(单词首字母大写)

类:QApplication

对于函数的命名 小驼峰法(,首个单词的首字母小写,后面的单词首字母大写)

setWindowTitle

# 4 .pro文件

```
在使用Qt向导生成的应用程序.pro文件格式如下:

QT       += core gui  //包含的模块
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets //大于Qt4版本 才包含widget模块
TARGET = QtFirst  //应用程序名  生成的.exe程序名称
TEMPLATE = app    //模板类型     应用程序模板
SOURCES += main.cpp\   //源文件
        mywidget.cpp
HEADERS  += mywidget.h   //头文件

 .pro就是工程文件(project)，它是qmake自动生成的用于生产makefile的配置文件。.pro文件的写法如下:
 1 注释
从“#”开始，到这一行结束。
```

```
l 模板变量告诉qmake为这个应用程序生成哪种makefile。下面是可供使用的选择：TEMPLATE = app
n app -建立一个应用程序的makefile。这是默认值，所以如果模板没有被指定，这个将被使用。
n lib - 建立一个库的makefile。
n vcapp - 建立一个应用程序的VisualStudio项目文件。
n vclib - 建立一个库的VisualStudio项目文件。
n subdirs -这是一个特殊的模板，它可以创建一个能够进入特定目录并且为一个项目文件生成makefile并且为它调用make的makefile。
l #指定生成的应用程序名：
TARGET = QtDemo
l #工程中包含的头文件
HEADERS += include/painter.h
l #工程中包含的.ui设计文件
FORMS += forms/painter.ui
l #工程中包含的源文件
SOURCES += sources/main.cpp sources
l #工程中包含的资源文件
RESOURCES += qrc/painter.qrc
l greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
这条语句的含义是，如果QT_MAJOR_VERSION大于4（也就是当前使用的Qt5及更高版本）需要增加widgets模块。如果项目仅需支持Qt5，
也可以直接添加“QT += widgets”一句。
不过为了保持代码兼容，最好还是按照QtCreator生成的语句编写。
l #配置信息
CONFIG用来告诉qmake关于应用程序的配置信息。
CONFIG += c++11 //使用c++11的特性
在这里使用“+=”，是因为我们添加我们的配置选项到任何一个已经存在中。这样做比使用“=”那样替换已经指定的所有选项更安全。
```

# 5 qt简单应用程序

```
main入口函数中
#include "widget.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Widget w;
    w.show();

```

```
    return a.exec();
}

```

解释：

l Qt系统提供的标准类名声明头文件没有.h后缀

l Qt一个类对应一个头文件，类名就是头文件名

l QApplication应用程序类

n 管理图形用户界面应用程序的控制流和主要设置。

n 是Qt的整个后台管理的命脉它包含主事件循环，在其中来自窗口系统和其它资源的所有事件处理和调度。它也处理应用程序的初始化和结束，并且提供对话管理。

n 对于任何一个使用Qt的图形用户界面应用程序，都正好存在一个QApplication 对象，而不论这个应用程序在同一时间内是不是有0、1、2或更多个窗口。

l a.exec()

程序进入消息循环，等待对用户输入进行响应。这里main()把控制权转交给Qt，Qt完成事件处理工作，当应用程序退出的时候exec()的值就会返回。在exec()中，

Qt接受并处理用户和系统的事件并且把它们传递给适当的窗口部件。