本节将介绍如何写入帧并创建视频文件。这样我们就完成了典型的视频处理过程：读取视频流，处理其中的帧，然后在新的视频文件中存储结果。

OpenCV用cv::VideoWriter类写视频文件。首先用VideoWriter类的open函数打开视频文件：

bool VideoWriter::open(const string& filename, // 文件名

int fourcc, // 所用的编解码器

double fps, // 视频的帧速率

Size frameSize, // 帧的尺寸

bool isColor=true)； // 彩色视频？

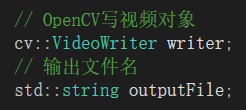
编解码器是一个软件模块，用于编码和解码视频流。编解码器定义了文件格式和存储信息的压缩方案。用某种编解码器进行编码的视频，必须用同一种编解码器才能解码。通常使用含有四个字符的代码来唯一地表示一种编解码器，例如CV\_FOURCC('P','I','M','1') 表示使用 MPEG-1 编解码器，完整的编解码器列表可参见：<http://www.fourcc.org/codecs.php>

在成功执行open后，即可通过调用VideoWriter类的write函数在文件中写入视频帧：

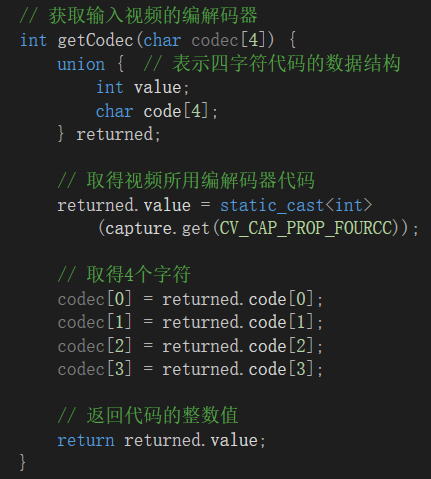
void VideoWriter::write(const Mat& image)

OK，让我们在11.3程序的基础上，开始编程吧！

1. 首先为VideoProcessor类增加2个新的成员属性：

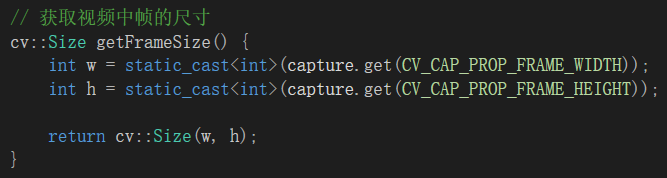


1. 首先添加成员函数，用于获取输入视频的编解码器：

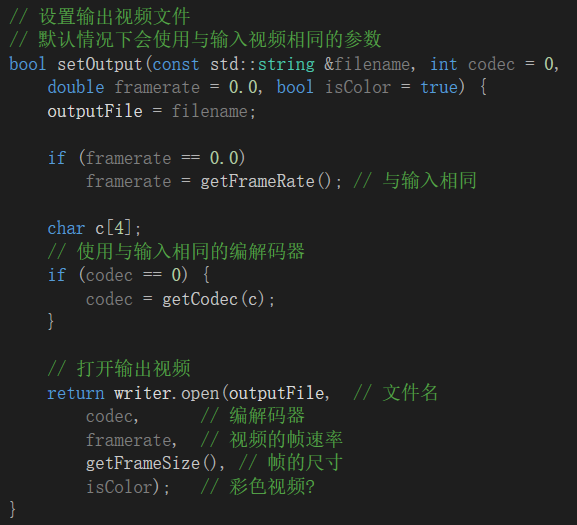


其中codec为输出参数，用于存储查询到的编解码器的字符形式，返回值为编解码器的整数形式

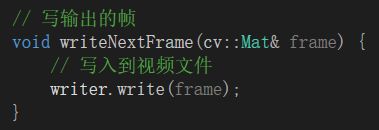
1. 接下来添加成员函数，用于获取视频中帧的尺寸：



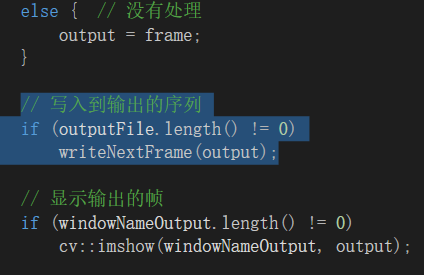
1. 添加成员函数，用以设置输出视频文件，默认情况下会使用与输入视频相同的参数：



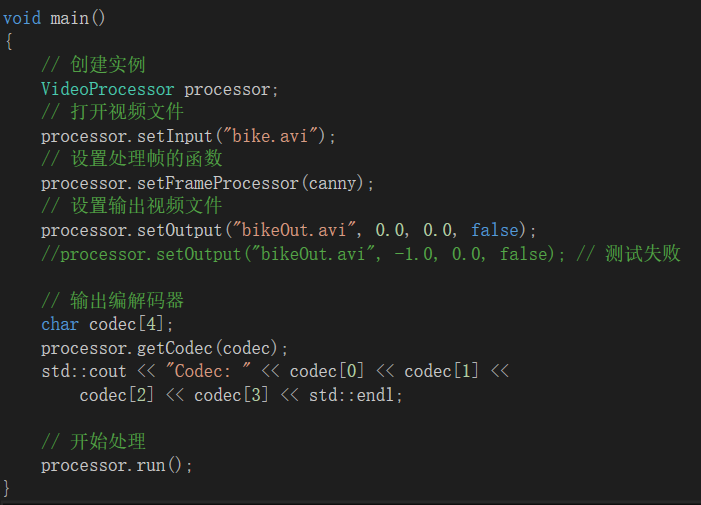
1. 然后添加私有方法，用以处理帧的写入过程：



1. 最后在run方法的视频捕获循环中添加一个新的步骤：



1. VideoProcessor的修改完成，可以在main函数中测试了：



说明：

* 1. 这里我们不设置输入和输出窗口以及帧间延迟，以提高测试效率；
  2. setOutput的中间2个参数（编解码器和帧速率）给0，即使用和输入视频相同的编解码器和帧速率；最后1个参数给false，因为canny算法的处理结果已非彩色图像；书上说编解码器给-1会弹出对话框让用户选择可用的编解码器，但昆哥在自己机器上并未测试成功，童鞋们可试一下
  3. 后面输出编解码器仅是为了查看一下输入视频所用的编解码器是什么^\_^

至此大功告成，童鞋们测试下，然后在项目文件夹下找到输出的视频文件，看看能否播放。