**任务**：从文件中读取一幅输入图像，在窗口中显示图像，应用一个处理函数，然后把输出图像存储到磁盘。

以下为演示和讲解部分：

1. 准备工作：创建一个空的控制台应用程序，并按照之前的方法设置项目属性，并添加一个cpp文件
2. 关于创建解决方案目录；
3. 关于通用项目设置，与仅针对本项目的项目属性设置
4. 头文件包含：

#include <opencv2/core/core.hpp> // 核心库

#include <opencv2/highgui/highgui.hpp> // GUI库

1. core.hpp可不用include，因为highgui.hpp中引用了core.hpp
2. 也可去掉中间目录core和highgui
3. .hpp与.h的区别？
4. 定义空图像

cv::Mat image; // 创建一个空图像(尺寸为0×0)

1. 关于cv命名空间
2. 笔记认识的OpenCV中的第一个数据结构（类）Mat
3. Mat的含义
4. 如何验证是空图像？
5. 输出图像的行数和列数

std::cout << "图像大小为 " << image.rows << " x "

<< image.cols << std::endl;

1. 使用std命名空间
2. 笔记认识的Mat的两个属性rows和cols(注意区分属性和方法)
3. 测试运行，闪退？Ctrl+F5试试

**学生自行验证以上内容**

1. 找一图像文件放到项目文件夹下，将其读入到程序：

image = cv::imread("puppy.jpg");

1. OpenCV所支持的图像格式
2. 笔记imread
3. imread的含义：image read
4. 这样写意味着图像放在当前目录，当前目录在哪里？
5. 如何判定读取是否成功？(读取失败不会报错)
6. 检查读取是否成功：

if (image.empty()) { // 图像为空？

std::cout << "读取图像失败!"<< std::endl;

}

1. 笔记Mat的empty方法
2. 改图像文件名测试一下
3. 直接运行exe测试下，什么原因？运行exe时当前目录不同
4. 定义窗口并显示图像：

cv::namedWindow("原始图像");

cv::imshow("原始图像", image);

1. 笔记namedWindow和imshow (意为image show)
2. 如果imshow中指定的窗口之前未定义，则会自动创建该窗口，因此namedWindow可以省略
3. 测试下，Ctrl+F5还管用么？Ctrl+F5阻挡不了程序结束（窗口销毁），只是命令控制台没关罢了
4. 新增按键等待函数，使得能看到窗口显示的图像：

cv::waitKey(0);

1. 笔记waitKey
2. 参数含义
3. 测试下，用不用Ctrl+F5已无区别

**学生自行验证以上内容**

1. 对图像进行处理，以将图像水平翻转为例：

cv::Mat result; // 创建另一个空图像用于存放处理结果

cv::flip(image, result, 1); // 水平翻转图像

1. 笔记flip
2. 参数含义，前两个参数看函数提示，第3个参数可以试
3. flip函数输出图像参数也可设为和输入图像相同，这样输入图像将被修改
4. 把处理结果show出来：

cv::imshow("输出图像", result);

1. 运行下看看，同时测试下flip第3个参数取不同值的情况
2. 把处理过的图像存储到磁盘里：

cv::imwrite("output.bmp", result);

1. 笔记imwrite
2. 图像的格式由文件名后缀决定
3. 测试下看看图像是否成功保存

**学生自行验证以上内容**

1. 在使用imread函数装载图像时，可以通过设置选项把它转为灰度图像，有些计算机视觉算法是必须使用灰度图像的

image = cv::imread("puppy.jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);

1. 笔记imread参数CV\_LOAD\_IMAGE\_GRAYSCALE
2. 测试效果
3. 其他参数有哪些？如何查看其他参数？  
   笔记CV\_LOAD\_IMAGE\_COLOR，默认情况
4. 可用channels方法检查图像的通道数(当编写适用于不同通道数量的图像处理程序时经常会用到)：

std::cout << "该图像有 " << image.channels() << " 通道" << std::endl;

1. 笔记Mat的channels方法
2. 测试效果(针对灰度图和彩色图)

切回PPT P22看温故实现原理部分，以及扩展部分：

扩展1：鼠标回调函数的使用

1. 笔记鼠标回调函数原型

void onMouse(

int event, // 事件类型

int x, int y, // 鼠标位置

int flags, // 事件发生时按下了鼠标哪个键

void \*param // 设置回调函数时指定的参数

);

1. Event取值示例：CV\_EVENT\_LBUTTONDOWN

Flags取值示例：CV\_EVENT\_FLAG\_LBUTTON  
通过F12得知其他取值可能

1. 实现鼠标左键单击时在命令控制台输出点击位置的像素值(暂时针对灰度图)。

第一步：编写鼠标回调函数（定义在main之前）

// 鼠标回调函数，实现点击输出像素值

// param存放的是Mat图像指针

void onMouse(int event, int x, int y, int flags, void \*param) {

Mat \*im = reinterpret\_cast<Mat\*>(param);

if (event == CV\_EVENT\_LBUTTONDOWN) { // 处理鼠标左键按下事件

cout << "坐标(" << x << ", " << y << ")处像素值为："

<< static\_cast<int>(im->at<uchar>(Point(x, y))) << endl;

}

}

1. reinterpret\_cast含义，此处可用强转代替
2. 此例中实际上可以不用switch，PPT上用的switch，给出要处理多种不同鼠标消息时的框架
3. 笔记Mat类at模版方法
4. 笔记Point结构
5. static\_cast的作用，其等价于强转

第二步：在main函数中设置鼠标回调函数

setMouseCallback("原始图像", onMouse, (void\*)&image);

1. 笔记setMouseCallback
2. 此处书上的reinterpret\_cast可以用强转代替

测试效果（注意针对灰度图）：

1. 测试下onMouse中的static\_cast不要是什么效果？
2. 图像的坐标系是怎样的？
3. 灰度图像素值的含义？
4. 如何支持彩色图像？(如有时间演示下)

扩展2：鼠标回调函数的使用

对照PPT P30-P32

笔记circle、ellipse、line、rectangle 以及 putText

让学生在main函数中加入上述绘图和写入文本的代码测试下

注意得重新调用imshow才能显示更新后的图像