实验3 使用数组

# 实验目的

掌握数组的使用方法;

数组的遍历和数组元素的操作;

动态分配一维和二维数组

# 实验内容

下面的代码已经将一幅图像“Fruits.jpg”读入到内存中。图像的大小是height行，width列像素，每个像素占一个字节，类型是unsigned char的整数（即在区间[0,255]内）。图像数据存储在一维指针img\_data指向的内存区，每一行所占的字节数是width\_step。因此，第i行第j列像素的下标是[width\_step\*i+j]。

请编程实现：

1. 判断width\_step是否等于列数\*每个元素所占字节数。
2. 动态分配二维数组a，将图像数据读入a中。即实现函数ReadImageData。
3. 将数组a的元素上下翻转。即第一行变为最后一行。即实现函数FlipImageUpDown。
4. 将变换后的数组a再写回img\_data所指的内存里。即实现函数WriteImageData。如果以上步骤正确，会显示翻转后的图像。
5. 将数组a的元素左右翻转。即第一列变为最后一列。即实现函数FlipImageLeftRight。
6. 将变换后的数组a再写回img\_data所指的内存里。即调用函数WriteImageData。如果以上步骤正确，会显示翻转后的图像。
7. 将图像缩小为原来尺寸的一半，存入动态分配内存的二维数组b。一个简单的做法是将a中的属于奇数行和奇数列的元素读取写入到b中。

#include "cv.h"

#include "highgui.h"

void ReadImageData(unsigned char \*src, int rows, int cols, int width\_step, unsigned char \*\*dst)

{

}

void WriteImageData(unsigned char \*\*src, int rows, int cols, int width\_step, unsigned char \*dst)

{

}

void FlipImageUpDown(unsigned char \*\*img, int rows, int cols)

{

}

void FlipImageLeftRight(unsigned char \*\*img, int rows, int cols)

{

}

int main(int argc, char\* argv[])

{

IplImage\* img = cvLoadImage("Fruits.jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);

cvNamedWindow("Image", CV\_WINDOW\_AUTOSIZE);*//创建窗口*

cvShowImage("Image", img);*//显示图像*

  cvWaitKey(0); *//等待按键*

unsigned char \*img\_data = (unsigned char \*)(img->imageData); *//读取图像*

int height = img->height; *//获取图像高度（行数）*

int width = img->width; *//获取图像宽度（列数）*

int width\_step = img->widthStep; *//获取图像每行所占的字节数*

// 判断width\_step是否等于列数\*每个元素所占字节数

// write your code here

// 声明二维数组a，大小是height\*width，unsigned char类型;

// write your code here

// 读取图像数据到二维数组a中，实现函数ReadImageData

// 比如这样调用ReadImageData（img\_data, height, width, width\_step, a）；

// write your code here

// 上下翻转图像

// write your code here

// 调用FlipImageUpDown

// write your code here

// 将数组a的数据写回源图像的内存区（img\_data指针所指的区域），即调用WriteImageData

// write your code here

cvShowImage("Image", img);

cvWaitKey(0); *//等待按键*

// 左右翻转图像

// 调用FlipImageLeftRight

// write your code here

// 调用WriteImageData

  cvShowImage("Image", img);

cvWaitKey(0); *//等待按键*

/// 动态分配二维数组b，用来存储缩小后的图像

// write your code here

/// 将原图缩小为原尺寸的一半，结果存入b中，添加相应的函数实现

// write your code here

IplImage \*img2 = cvCreateImage(cvSize(width/2, height/2), IPL\_DEPTH\_8U, 1);

WriteImageData(b, img2->width, img2->height, img2->widthStep, (unsigned char\*)(img2->imageData));

cvShowImage("Image", img2);

cvWaitKey(0); *//等待按键*

cvDestroyWindow("Image");*//销毁窗口*

cvReleaseImage(&img); *//释放图像*

cvReleaseImage(&img2); *//释放图像*

// 释放二维数组a和b的空间

// write your code here

return 0;

}

# 实验要求

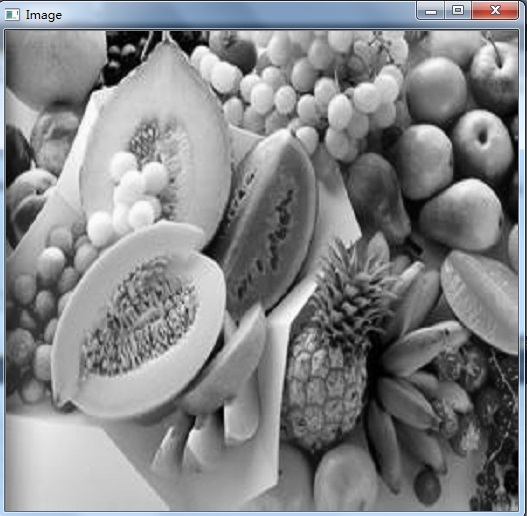
完成上述代码，并能显示正确的结果图像。

# 注意事项

1. 图像的基本组成单元用“像素”表示，例如如果图像的大小是480\*511像素，表示图像的高度（行数）为480像素，宽度（列数）为511像素，；
2. 本实例中加载图像的时候将其处理为灰度图像，每个像素在内存中占一个字节。
3. 二维数组作为函数参数时，应在函数声明时，指明其行列数，否则程序内无法得知数组的维度；
4. 动态分配的数组，使用完后要及时释放，防止内存的泄漏。

# 实验效果图(仅供参考)

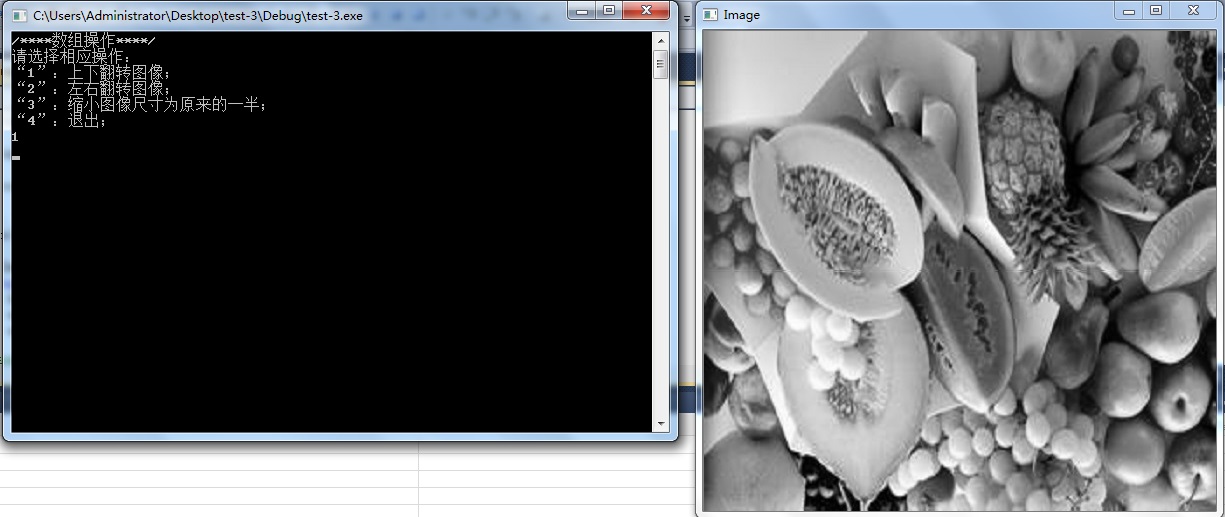
1. 原图像



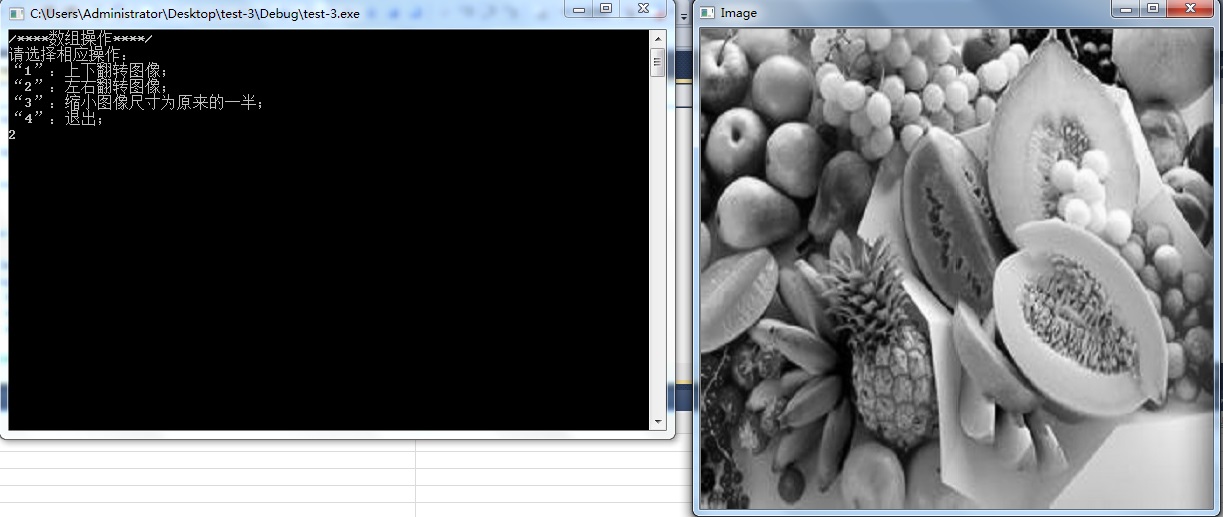
1. 程序主界面



1. 图像上下翻转



1. 图像左右翻转



1. 图像尺寸缩小为原来的一半

