## 1 读取文件名

### #if……#endif

#if 0

code

#endif

(1)code中定义的是一些调试版本的代码，此时code完全被编译器忽略。如果想让code生效，只需把#if 0改成#if 1

(2)#if 0还有一个重要的用途就是用来当成注释，如果你想要注释的程序很长，这个时候#if 0是最好的，保证不会犯错误。（但是林锐的书上说千万不要把#if 0 来当作块注释使用） #if 1可以让其间的变量成为局部变量。

(3)这个结构表示你先前写好的code，现在用不上了，又不想删除，就用这个方法，比注释方便。

### #ifdef \_WIN32……#endif

一般情况下，源程序中所有的行都参加编译。但是有时希望对其中一部分内容只在满足一定条件才进行编译，也就是对一部分内容指定编译的条件，这就是“条件编译”。有时，希望当满足某条件时对一组语句进行编译，而当条件不满足时则编译另一组语句。

条件编译命令最常见的形式为：

#ifdef 标识符

程序段1

#else

程序段2

#endif

它的作用是：当标识符已经被定义过(一般是用#define命令定义)，则对程序段1进行编译，否则编译程序段2。

其中#else部分也可以没有，即：

#ifdef

程序段1

#denif

### string的相关操作

### ifndef/#define/#endif

#ifndef A\_H意思是"if not define a.h"  如果不存在a.h，接着的语句应该#define A\_H  就引入a.h，最后一句应该写#endif   否则不需要引

## 图像处理部分



### 2.1 colorMatch

//颜色匹配

//! 根据一幅图像与颜色模板获取对应的二值图

//! 输入RGB图像, 颜色模板（蓝色、黄色）

//! 输出灰度图（只有0和255两个值，255代表匹配，0代表不匹配）

### sobelyuchuli

均值滤波→sobel边缘检测→二值化→闭运算

### findContours

### vector<vector<Point> >::iterator

### RotatedRect

class CV\_EXPORTS RotatedRect

{

public:

*//构造函数*

RotatedRect();

RotatedRect(const Point2f& center, const Size2f& size, float angle);

RotatedRect(const CvBox2D& box);

*//!返回矩形的4个顶点*

void points(Point2f pts[]) const;

*//返回包含旋转矩形的最小矩形*

Rect boundingRect() const;

*//!转换到旧式的cvbox2d结构*

operator CvBox2D() const;

Point2f center; *//矩形的质心*

Size2f size; *//矩形的边长*

float angle; *//旋转角度，当角度为0、90、180、270等时，矩形就成了一个直立的矩形*

}

### minAreaRect

RotatedRect minAreaRect(InputArray **points**)

### *CV\_EXPORTS\_W* *Rect* *boundingRect*( *InputArray* points );

### template<typename \_Tp> class Rect\_

template<typename \_Tp> class *Rect\_*

{

public:

typedef \_Tp *value\_type*;

//! various constructors

*Rect\_*();

*Rect\_*(\_Tp \_x, \_Tp \_y, \_Tp \_width, \_Tp \_height);

*Rect\_*(const *Rect\_*& r);

*Rect\_*(const *CvRect*& r);

*Rect\_*(const *Point\_*<\_Tp>& org, const *Size\_*<\_Tp>& sz);

*Rect\_*(const *Point\_*<\_Tp>& pt1, const *Point\_*<\_Tp>& pt2);

*Rect\_*& operator = ( const *Rect\_*& r );

//! the top-left corner

*Point\_*<\_Tp> *tl*() const;

//! the bottom-right corner

*Point\_*<\_Tp> *br*() const;

//! size (width, height) of the rectangle

*Size\_*<\_Tp> size() const;

//! area (width\*height) of the rectangle

\_Tp area() const;

//! conversion to another data type

template<typename \_Tp2> operator *Rect\_*<\_Tp2>() const;

//! conversion to the old-style CvRect

operator *CvRect*() const;

//! checks whether the rectangle contains the point

bool *contains*(const *Point\_*<\_Tp>& pt) const;

\_Tp x, y, width, height; //< the top-left corner, as well as width and height of the rectangle

};

### boundingRect

class *CV\_EXPORTS* *RotatedRect*

{

public:

//! various constructors

*RotatedRect*();

*RotatedRect*(const *Point2f*& center, const *Size2f*& size, float angle);

*RotatedRect*(const *CvBox2D*& box);

//! returns 4 vertices of the rectangle

void points(*Point2f* pts[]) const;

//! returns the minimal up-right rectangle containing the rotated rectangle

*Rect* *boundingRect*() const;

//! conversion to the old-style CvBox2D structure

operator *CvBox2D*() const;

*Point2f* center; //< the rectangle mass center

*Size2f* size; //< width and height of the rectangle

float angle; //< the rotation angle. When the angle is 0, 90, 180, 270 etc., the rectangle becomes an up-right rectangle.

};

### template<typename \_Tp> class Rect\_

### template<typename \_Tp> class Rect\_

### template<typename \_Tp> class Rect\_

### template<typename \_Tp> class Rect\_