使用说明书

image_t* ReadPNM(char* input)	读取 PNM 文件，支持 PBM、PGM 和 PPM 图像。
	需引入以下结构体：
	typedef struct color_t {
	  uint8_t r; //Red
	  uint8_t g; //Green
	  uint8_t b; //Blue
	  uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	  color_t c; //RGBA
	  uint8_t g; //灰度
	  uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //宽
	  uint32_t height; //高
	  uint16_t color_type; //颜色种类
	  uint16_t palette_num; //调色板数量
	  color_t *palette; //指向调色板的指针
	  pixcel_t **map; //图像数据
	} image_t;

void WritePNM(image_t* input,char* output,int type)	PNM 图像数据保存为图像文件，支持 PBM、PGM 和 PPM 图像，type 是 PNM 文件的格式，如 type=1、2、3、4、5、6。
	需引入以下结构体：
	typedef struct color_t {
	  uint8_t r; //Red
	  uint8_t g; //Green
	  uint8_t b; //Blue
	  uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	  color_t c; //RGBA
	  uint8_t g; //灰度
	  uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //宽
	  uint32_t height; //高
	  uint16_t color_type; //颜色种类

	uint16_t palette_num; //调色板数量
	color_t *palette; //指向调色板的指针
	pixcel_t **map; //图像数据
	} image_t;

image_t* ReadBMP(char* input)	读取 BMP 图像。
	需引入以下结构体：
	typedef struct color_t {
	uint8_t r; //Red
	uint8_t g; //Green
	uint8_t b; //Blue
	uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	color_t c; //RGBA
	uint8_t g; //灰度
	uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {
	uint32_t width; //宽
	uint32_t height; //高
	uint16_t color_type; //颜色种类
	uint16_t palette_num; //调色板数量
	color_t *palette; //指向调色板的指针
	pixcel_t **map; //图像数据
	} image_t;

void WriteBMP(image_t* input,char* output,int compress)	BMP 图像数据保存为图像文件，compress=1 时进行 RLE 压缩。
	需引入以下结构体：
	typedef struct color_t {
	uint8_t r; //Red
	uint8_t g; //Green
	uint8_t b; //Blue
	uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	color_t c; //RGBA
	uint8_t g; //灰度
	uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {

	uint32_t width; //宽
	uint32_t height; //高
	uint16_t color_type; //颜色种类
	uint16_t palette_num; //调色板数量
	color_t *palette; //指向调色板的指针
	pixcel_t **map; //图像数据
	} image_t;

void WriteBMP(image_t* input,char* output)	BMP 图像数据保存为图像文件。
	需引入以下结构体：
	typedef struct color_t {
	uint8_t r; //Red
	uint8_t g; //Green
	uint8_t b; //Blue
	uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	color_t c; //RGBA
	uint8_t g; //灰度
	uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {
	uint32_t width; //宽
	uint32_t height; //高
	uint16_t color_type; //颜色种类
	uint16_t palette_num; //调色板数量
	color_t *palette; //指向调色板的指针
	pixcel_t **map; //图像数据
	} image_t;

void WriteBMP1(image_t* input,char* output,int compress)	BMP 图像数据保存为图像文件，compress=1 时进行 RLE 压缩。
	需引入以下结构体：
	typedef struct color_t {
	uint8_t r; //Red
	uint8_t g; //Green
	uint8_t b; //Blue
	uint8_t a; //Alpha
	} color_t;
	typedef union pixcel_t {
	color_t c; //RGBA
	uint8_t g; //灰度

	uint8_t i; //颜色索引
	} pixcel_t;
	typedef struct image_t {
	uint32_t width; //宽
	uint32_t height; //高
	uint16_t color_type; //颜色种类
	uint16_t palette_num; //调色板数量
	color_t *palette; //指向调色板的指针
	pixcel_t **map; //图像数据
	} image_t;

void ImageFusion(char* input1,char* input2,char* output,int block_height,int block_width,double threshold)	多聚焦图像的融合,支持 8 位 BMP 图像。block_height=8 , block_width=8 , threshold=1.75。
void ImageFusion(char* input1,char* input2,char* MaskImage,char* output,int dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS)	图像融合。参考：a=3，b1=4，DX1=-68，DY1=-99，EPS=1，input1="图像融合 1.jpg"，input2="图像融合 2.jpg"，MaskImage="掩膜.png"，output="output.jpg"。 int dx[] = {0,0,-1,1}; int dy[] = {-1,1,0,0};
void ImageFusion(char* input1,char* input2,char* inputUniqe1,char* inputUniqe2,char* output)	图像融合，支持 PNG 图像。参考：input1="图像融合 1.png"，input2="图像融合 2.png"，inputUniqe1="图像融合 1_unique.txt"，inputUniqe2="图像融合 2_unique.txt"。
void Uniqe(char* input,char* inputUniqe,char* output,double R,double G,double B)	图像融合，支持 PNG 图像。参考：input="图像融合 1.png"，inputUniqe="图像融合 1_unique.txt"。R=255，G=0，B=0。
void Screenshot1(HWND hWnd,LPCWSTR OutputImage)	截屏函数。hWnd 是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。
void Screenshot2(HWND hWnd,LPCWSTR OutputImage)	截屏函数。hWnd 是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。
void Screenshot3(HWND hWnd,LPCWSTR OutputImage)	截屏函数。hWnd 是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。
uint8_t* AESencrypt(uint8_t* input,uint8_t* key,int size)	AES 加密函数，input 是原数据，key 是密钥，size 是 input 的大小。返回加密

	结果数据。
uint8_t* AESdecrypt(uint8_t* input,uint8_t* key,int size)	AES 解密函数,input 是已加密数据,key 是密钥，size 是 input 的大小。返回解密结果数据。
void DES_Encrypt(char *PlainFile, char *Key,char *CipherFile)	DES 加密函数，支持多种文件。PlainFile 是原文件的文件名,Key 是密钥字符，CipherFile 是加密后的文件名。
void DES_Decrypt(char *CipherFile, char *Key,char *PlainFile)	DES 解密函数，支持多种文件。CipherFile 是已加密文件的文件名，Key 是密钥字符,PlainFile 是解密后的文件名。
int Equal(char* input1,char* input2,double c)	若比对图像的梯度幅相似性偏差值等于 c 则通过。input1 和 input2 是要比对的两个图像。c 是参考的阈值。支持 24 位 BMP 图像。
int GreaterThan(char* input1,char* input2,double c)	若比对图像的梯度幅相似性偏差值大于 c 则通过。input1 和 input2 是要比对的两个图像。c 是参考的阈值。支持 24 位 BMP 图像。
int LessThan(char* input1,char* input2,double c)	若比对图像的梯度幅相似性偏差值小于 c 则通过。input1 和 input2 是要比对的两个图像。c 是参考的阈值。支持 24 位 BMP 图像。
double GMSD(char* input1, char* input2)	求两幅图像的梯度幅相似性偏差值并返回结果。input1 和 input2 是要比对的两个图像。支持 24 位 BMP 图像。
void FileWrite(char* BMP,char* TXT)	图像隐写之文件写入，将文本文件写入图像。支持 32 位 BMP 图像。BMP 是要写入的图像文件名，TXT 是要写入图像的文本文件名。
void FileWriteOut(char* BMP,char* TXT)	图像隐写之文件写出，将文本文件从图像中取出来。支持 32 位 BMP 图像。BMP 是要写出的图像文件名，TXT 是写出图像后信息保存的文本文件名。
void Watershed2(char* input,char* inputMarqueurs,char* output,int r,unsigned char R,unsigned char G,unsigned char B)	图像分割之分水岭算法。inputMarqueurs 是输入图像的标记图像。R=230，G=0，B=0，r=1。支持 24 位 BMP 图像。
void EcrireImage1(char* input,char* output,uint32_t rayon)	图像分割。rayon=5。支持 24 位 BMP 图像。
void EcrireImage2(char*	图像分割。rayon=5。支持 24 位 BMP 图

input,char* inputMarqueurs,char* output,uint32_t rayon)	像。
void EcrireLPECouleur1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	图像分割。rayon=5。支持 24 位 BMP 图像。
void Watershed1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	图像分割之分水岭算法。inputMarqueurs 是输入图像的标记图像。rayon=5。支持 24 位 BMP 图像。
void EcrireImage3(char* input,char* inputMarqueurs,char* output,uint16_t rayon)	图像分割。rayon=1。支持 24 位 BMP 图像。
void EcrireImageCouleursAleatoires(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint16_t rayon)	图像分割。rayon=1。支持 24 位 BMP 图像。
void Watershed(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint8_t a,uint16_t rayon)	图像分割之分水岭算法。inputMarqueurs 是输入图像的标记图像。a 一般为 255，rayon=1。支持 24 位 BMP 图像。
double CharacterRecognition(char* TargetImage,char* TemplateFileGroup[])	字符匹配，支持 BMP 图像，返回值是目标图像匹配到的模板文件的序号，如返回值是 2 则说明图像与序号为 2（序号从零开始）的模板匹配。 参考： TemplateFileGroup[]={"0.txt", "1.txt", "2.txt", "3.txt", "4.txt", "5.txt", "6.txt", "7.txt", "8.txt", "9.txt" };
double CharacterRecognition1(char* TargetImage,char* TemplateFileGroup[])	字符匹配，支持 BMP 图像，返回值是目标图像匹配到的模板文件的序号，如返回值是 2 则说明图像与序号为 2（序号从零开始）的模板匹配。 参考： TemplateFileGroup[]={"0.txt", "1.txt", "2.txt", "3.txt", "4.txt", "5.txt", "6.txt",

	"7.txt", "8.txt", "9.txt" };
void CharacterSegmentation(char* input, string OutputFolder, int YHistogramValleyMaxPixelNumber, int XHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int SingleNumberImgBoundary, int Infinite, double NumberImageBlackPixelPercentage)	字符分割。支持 BMP 图像。OutputFolder 是结果输出的文件夹，如“output”，输出结果的文件名的构成方式为:左上角的 X 坐标-左上角的 Y 坐标-右下角的 X 坐标-右下角的 Y 坐标，YHistogramValleyMaxPixelNumber 是求 Y 方向直方图，谷的最少黑色像素个数，YHistogramValleyMaxPixelNumber=0，XHistogramValleyMaxPixelNumber 是求 X 方向直方图，谷的最少黑色像素个数，XHistogramValleyMaxPixelNumber=4，SubImgBlackPixelPercentage 是一张子图内黑色像素超过一定百分比才算有数字，SubImgBlackPixelPercentage=0.001，SingleNumberImgBoundary 是单张数字图像边缘填充宽度，SingleNumberImgBoundary=5,Infinite 视作无穷大，Infinite=249480，NumberImageBlackPixelPercentage 是单张数字图像黑色像素个数超过所有数字图像，NumberImageBlackPixelPercentage=0.35。
void CharacterSegmentation(char* input,char* output, int BoundaryRemoveGap, int BinaryGap,int YHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int Infinite, int XHistogramValleyMaxPixelNumber, double NumberImageBlackPixelPercentage, int SingleNumberImgBoundary)	字符分割。支持 BMP 图像。BinaryGap 是图像二值化全局阈值，BinaryGap=135，BoundaryRemoveGap 是边缘全设为白色的距离，BoundaryRemoveGap=7，Infinite 是视作无穷大，Infinite=249480，SingleNumberImgBoundary 是单张数字图像边缘填充宽度，SingleNumberImgBoundary=5，YHistogramValleyMaxPixelNumber 是求 Y 方向直方图，谷的最少黑色像素个数，YHistogramValleyMaxPixelNumber=0，XHistogramValleyMaxPixelNumber 是求 X 方向直方图，谷的最少黑色像素个数，XHistogramValleyMaxPixelNumber=4，

	SubImgBlackPixelPercentage 是一张子图内黑色像素超过一定百分比才算有数字，SubImgBlackPixelPercentage=0.001，NumberImageBlackPixelPercentage 是单张数字图像黑色像素个数超过所有数字图像，NumberImageBlackPixelPercentage=0.35。 参考：output="output"。
void CodeEncoding(std::string input,char* output, int width,int height, int margin, int eccLevel, int stride_bytes, int comp,int a)	二维码编码。input 是要编码的字符串，output 是生成的二维码图像文件名。 margin:条形码周围的边距 ecc：纠错级别，[0-8] a=1：AZTEC a=2：CODABAR a=3：CODE_39 a=4：CODE_93 a=5：CODE_128 a=6：DATA_MATRIX a=7：EAN_8 a=8：EAN_13 a=9：ITF a=10：MAXICODE a=11：PDF_417 a=12：QR_CODE a=13：RSS_14 a=14：RSS_EXPANDED a=15：UPC_A a=16：UPC_E a=17：UPC_EAN_EXTENSION 参考：margin=10，eccLevel=-1，stride_bytes=0，comp=1。
std::string CodeDecoding(char* input,int req_comp,int a)	二维码解码。input 是输入的二维码图像文件名，返回解码结果。 a=1：Lum a=2：RGB a=3：BGR a=4：RGBX a=5：XRGB a=6：BGRX a=7：XBGR 参考：req_comp=4，a=4。