Instrucciones de uso

image_t* ReadPNM(char* input)	Leer archivos pnm, soporte para imágenes PBM, PGM y PPM.
	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises
	  uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //Ancho
	  uint32_t height; //Alto
	  uint16_t color_type; //Tipo de color
	  uint16_t palette_num; //Número de paletas
	  color_t *palette; //Puntero a la paleta de colores
	  pixcel_t **map; //Datos de imagen
	} image_t;

void WritePNM(image_t* input,char* output,int type)	Los datos de imagen PNM se guardan como archivos de imagen, admitiendo imágenes PBM, PGM y PPM, y el tipo es el formato de archivo pnm, como tipo = 1, 2, 3, 4, 5, 6.
	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises

	uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	uint32_t width; //Ancho
	uint32_t height; //Alto
	uint16_t color_type; //Tipo de color
	uint16_t palette_num; //Número de paletas
	color_t *palette; //Puntero a la paleta de colores
	pixcel_t **map; //Datos de imagen
	} image_t;

image_t* ReadBMP(char* input)	Lee la imagen BMP.
	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises
	uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	uint32_t width; //Ancho
	uint32_t height; //Alto
	uint16_t color_type; //Tipo de color
	uint16_t palette_num; //Número de paletas
	color_t *palette; //Puntero a la paleta de colores
	pixcel_t **map; //Datos de imagen
	} image_t;

void WriteBMP(image_t* input,char* output,int compress)	Los datos de imagen BMP se guardan como archivos de imagen, mientras que compress = 1 realiza compresión rle.

	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises
	  uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //Ancho
	  uint32_t height; //Alto
	  uint16_t color_type; //Tipo de color
	  uint16_t palette_num; //Número de paletas
	  color_t *palette; //Puntero a la paleta de colores
	  pixcel_t **map; //Datos de imagen
	} image_t;

void WriteBMP(image_t* input,char* output)	Los datos de imagen BMP se guardan como archivos de imagen.
	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises
	  uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //Ancho
	  uint32_t height; //Alto
	  uint16_t color_type; //Tipo de

	color
	  uint16_t palette_num; //Número de paletas
	  color_t *palette; //Puntero a la paleta de colores
	  pixcel_t **map; //Datos de imagen
	} image_t;

void WriteBMP1(image_t* input,char* output,int compress)	Los datos de imagen BMP se guardan como archivos de imagen, mientras que compress = 1 realiza compresión rle.
	Es necesario introducir las siguientes estructuras:
	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; //Escala de grises
	  uint8_t i; //Índice de color
	} pixcel_t;
	typedef struct image_t {
	  uint32_t width; //Ancho
	  uint32_t height; //Alto
	  uint16_t color_type; //Tipo de color
	  uint16_t palette_num; //Número de paletas
	  color_t *palette; //Puntero a la paleta de colores
	  pixcel_t **map; //Datos de imagen
	} image_t;

void ImageFusion(char* input1,char* input2,char* output,int block_height,int block_width,double threshold)	Fusión de imágenes multifocales, soporte de imágenes BMP de 8 dígitos. block_height = 8, block_width = 8, threshold = 1,75.
void ImageFusion(char* input1,char* input2,char* MaskImage,char* output,int	Fusión de imágenes. Referencia: a=3, b1=4, DX1=-68, DY1=-99, EPS=1, input1=" Fusión de imágenes 1.jpg",

dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS)	input2=″Fusión de imágenes 2.jpg″, MaskImage=″ Mascarilla.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)	Fusión de imágenes, soporte de imágenes png. Referencia：input1=” Fusión de imágenes 1.png” , input2=” Fusión de imágenes 2.png”, inputUniqe1=” Fusión de imágenes 1_unique.txt” , inputUniqe2=” Fusión de imágenes 2_unique.txt”。
void Uniqe(char* input,char* inputUniqe,char* output,double R,double G,double B)	Fusión de imágenes, soporte de imágenes png. Referencia：input=” Fusión de imágenes 1.png” , inputUniqe=” Fusión de imágenes 1_unique.txt”。R=255, G=0, B=0。
void Screenshot1(HWND hWnd, LPCWSTR OutputImage)	Función de captura de pantalla. Hwnd es el mango de la ventana para tomar una captura de pantalla, como : GetDesktopWindow() ; OutputImage es el nombre de la captura de pantalla.
void Screenshot2(HWND hWnd,LPCWSTR OutputImage)	Función de captura de pantalla. Hwnd es el mango de la ventana para tomar una captura de pantalla, como : GetDesktopWindow() ; OutputImage es el nombre de la captura de pantalla.
void Screenshot3(HWND hWnd, LPCWSTR OutputImage)	Función de captura de pantalla. Hwnd es el mango de la ventana para tomar una captura de pantalla, como : GetDesktopWindow() ; OutputImage es el nombre de la captura de pantalla.
uint8_t* AESencrypt(uint8_t* input,uint8_t* key,int size)	Función de cifrado aes, input es el dato original, key es la clave y size es el tamaño de input. Devuelve los datos del resultado cifrado.
uint8_t* AESdecrypt(uint8_t* input,uint8_t* key,int size)	Función de descifrado aes, input es datos encriptados, key es la clave y size es el tamaño de input.

	Devuelve los datos del resultado descifrado.
void DES_Encrypt(char *PlainFile, char *Key,char *CipherFile)	Función de cifrado DES, compatible con una variedad de archivos. PlainFile es el nombre del archivo original, Key es el carácter clave y CipherFile es el nombre del archivo cifrado.
void DES_Decrypt(char *CipherFile, char *Key,char *PlainFile)	La función de descifrado des admite una variedad de archivos. CipherFile es el nombre del archivo encriptado, Key es el carácter clave y PlainFile es el nombre del archivo descifrado.
int Equal(char* input1,char* input2,double c)	Si el valor de desviación de similitud de amplitud de gradiente de la imagen de comparación es igual a c, pasa. Entrada 1 e Entrada 2 son las dos imágenes a comparar. c es el umbral de referencia. Admite imágenes BMP de 24 bits.
int GreaterThan(char* input1,char* input2,double c)	Si el valor de desviación de similitud de la amplitud del gradiente de la imagen de comparación es mayor que c, se pasa. Entrada 1 e Entrada 2 son las dos imágenes a comparar. c es el umbral de referencia. Admite imágenes BMP de 24 bits.
int LessThan(char* input1,char* input2,double c)	Si el valor de desviación de similitud de la amplitud del gradiente de la imagen de comparación es inferior a c, se pasa. Entrada 1 e Entrada 2 son las dos imágenes a comparar. c es el umbral de referencia. Admite imágenes BMP de 24 bits.
double GMSD(char* input1, char* input2)	Encontrar el valor de desviación de similitud de amplitud de gradiente de las dos imágenes y devolver el resultado. Entrada 1 e Entrada 2 son las dos imágenes a comparar. Admite imágenes BMP de 24 bits.

void FileWrite(char* BMP,char* TXT)	Se escribe un archivo oculto de la imagen, y se escribe un archivo de texto en la imagen. Admite imágenes BMP de 32 bits. BMP es el nombre del archivo de imagen a escribir, y txt es el nombre del archivo de texto a escribir en la imagen.
void FileWriteOut(char* BMP,char* TXT)	Se escribe un archivo oculto de la imagen, que extrae el archivo de texto de la imagen. Admite imágenes BMP de 32 bits. BMP es el nombre del archivo de imagen a escribir, y txt es el nombre del archivo de texto guardado por la información después de escribir la imagen.
void Watershed2(char* input,char* inputMarqueurs,char* output,int r,unsigned char R,unsigned char G,unsigned char B)	Algoritmo de cuenca para la segmentación de imágenes. inputMarqueurs es una imagen marcada de la imagen introducida. R = 230, G = 0, B = 0, r = 1. Admite imágenes BMP de 24 bits.
void EcrireImage1(char* input,char* output,uint32_t rayon)	Segmentación de imágenes. rayon = 5. Admite imágenes BMP de 24 bits.
void EcrireImage2(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentación de imágenes. rayon = 5. Admite imágenes BMP de 24 bits.
void EcrireLPECouleur1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentación de imágenes. rayon = 5. Admite imágenes BMP de 24 bits.
void Watershed1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Algoritmo de cuenca para la segmentación de imágenes. inputMarqueurs es una imagen marcada de la imagen introducida. rayon = 5. Admite imágenes BMP de 24 bits.
void EcrireImage3(char* input,char* inputMarqueurs,char* output,uint16_t rayon)	Segmentación de imágenes. rayon = 1. Admite imágenes BMP de 24 bits.
void EcrireImageCouleursAleatoires(c	Segmentación de imágenes. rayon = 1. Admite imágenes BMP de 24 bits.

har* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint16_t rayon)	
void Watershed(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint8_t a,uint16_t rayon)	Algoritmo de cuenca para la segmentación de imágenes. Inputmarqueurs es una imagen marcada de la imagen introducida. a generalmente es 255, rayon = 1. Admite imágenes BMP de 24 bits.
double CharacterRecognition(char* TargetImage,char* TemplateFileGroup[])	Coincidencia de caracteres, soporte para imágenes bmp, el valor de retorno es el número de serie del archivo de plantilla al que coincide la imagen objetivo, si el valor de retorno es 2, significa que la imagen coincide con la plantilla con el número de serie 2 (el número de serie comienza en cero). Referencia : 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[])	Coincidencia de caracteres, soporte para imágenes bmp, el valor de retorno es el número de serie del archivo de plantilla al que coincide la imagen objetivo, si el valor de retorno es 2, significa que la imagen coincide con la plantilla con el número de serie 2 (el número de serie comienza en cero). Referencia : 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,	División de caracteres. Admite imágenes bmp. OutputFolder es la carpeta a la que se exportan los resultados, como

int XHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int SingleNumberImgBoundary, int Infinite, double NumberImageBlackPixelPercentage)	"output", y el nombre del archivo que exporta los resultados se compone de: la coordenada X en la esquina superior izquierda - la coordenada y en la esquina superior izquierda - la coordenada X en la esquina inferior derecha - la coordenada y en la esquina inferior derecha , YHistogramValleyMaxPixelNumber es el número mínimo de píxeles negros en el valle para encontrar el histograma de dirección Y , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber es encontrar el histograma de dirección x, el número mínimo de píxeles negros en el valle , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage es un subinforme en el que los píxeles negros superan un cierto porcentaje para tener números , SubImgBlackPixelPercentage=0.001, SingleNumberImgBoundary es el ancho de relleno del borde de una sola imagen digital , SingleNumberImgBoundary=5,Infinite considera infinito , Infinite=249480 , NumberImageBlackPixelPercentage es una sola imagen digital con más píxeles negros que todas las imágenes digitales , NumberImageBlackPixelPercentage=0.35。
void CharacterSegmentation(char* input,char* output, int BoundaryRemoveGap, int BinaryGap,int YHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int	División de caracteres. Admite imágenes bmp. BinaryGap es el umbral global de la binarización de la imagen , BinaryGap=135,BoundaryRemoveGap es una distancia en la que los bordes están todos establecidos en blanco, BoundaryRemoveGap=7 , Infinite se

Infinite, int XHistogramValleyMaxPixelNumber, double NumberImageBlackPixelPercentage, int SingleNumberImgBoundary)	considera infinito , Infinite=249480 , SingleNumberImgBoundary es el ancho de relleno del borde de una sola imagen digital , SingleNumberImgBoundary=5 , YHistogramValleyMaxPixelNumber es el número mínimo de píxeles negros en el valle para encontrar el histograma de dirección Y , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber es encontrar el histograma de dirección x, el número mínimo de píxeles negros en el valle , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage es un subinforme en el que los píxeles negros superan un cierto porcentaje para tener números , SubImgBlackPixelPercentage=0.001, NumberImageBlackPixelPercentage es una sola imagen digital con más píxeles negros que todas las imágenes digitales , NumberImageBlackPixelPercentage=0.35。 Referencia: output="output"。
void CodeEncoding(std::string input,char* output, int width,int height, int margin, int eccLevel, int stride_bytes, int comp,int a)	Codificación de código qr. input es la cadena a codificar y output es el nombre del archivo de imagen de código QR generado. margin: márgenes alrededor del Código de barras ecc: nivel de corrección de errores, [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)	Decodificación de código qr. input es el nombre de archivo de imagen de código QR introducido, que devuelve el resultado de la decodificación. 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。