

嵌入式程式C/C++

- C/C++
 - 結構、類別
 - Bmp 影像格式 → 自建影像讀取類別
- Qt
 - QImage類別
 - 圖像顯示元件使用
 - 影像資料應用
 - 使用自建影像讀取類別進行影像檔案資料管理

嵌入式程式C/C++

- 結構(struct)：
- struct 是 C 中用來組織資料的關鍵字，為一種資料型態。
- 可自行設計一組資料結構，資料間有關聯性卻存在不同型態。
- 結構中可內鑲另一結構，但不能內鑲相同結構

Example 1

```
#include <stdio.h>
#include <string.h>

struct Ball {
    char color[10];
    double radius;
};

int main(void) {
    struct Ball ball1 = {"red", 5.0};

    struct Ball ball2;
    strcpy(ball2.color, "green");
    ball2.radius = 10.0;

    printf("ball1: %s,\t%.2f\n", ball1.color, ball1.radius);
    printf("ball2: %s,\t%.2f\n", ball2.color, ball2.radius);

    return 0;
}
```

Example 2

```
#include <stdio.h>
#include <string.h>

struct ImageSize
{
    int wid;
    int hei;
};

struct ImageFormat
{
    char format[10];
    ImageSize size;
    int length;
};

int main(void) {

    ImageFormat imgfmt;

    imgfmt.format = "BMP";
    imgfmt.size.wid = 100;
    imgfmt.size.hei = 50;
    imgfmt.length = imgfmt.size.wid * imgfmt.size.hei;

    printf("format: %s,\t%d\n,\t%d\n,\t%d\n", imgfmt.format, imgfmt.size.wid,
        imgfmt.size.hei, imgfmt.length);

    return 0;
}
```

Embedded Software Porting

- 類別 (Class) :

Class是C++中用來封裝資料的關鍵字，當您使用類別來定義一個物件 (Object) 時，您考慮這個物件可能擁有的「屬性」 (Property) 與「方法」 (Method) 。

- **屬性**是物件的靜態**描述**
- **方法**是物件上的動態**操作**。
- 在類別封裝時，有一個基本原則是：資訊的最小化公開，是基於安全性的考量，避免程式設計人員隨意操作屬性成員而造成程式的錯誤。

public 公開 VS private 私有

Embedded Software Porting

first_class.h

```
#pragma once
class Area_Calculator
{
public:
    Area_Calculator();
    ~Area_Calculator();

private:
    double m_radius;

public:
    void setRadius(double a_radius);
    double GetArea();
};
```

→ 公開方法
建構子、解構子

→ 私有屬性

→ 公開方法

first_class.cpp

```
#include "first_class.h"

Area_Calculator::Area_Calculator()
{
    m_radius = area = 0;
}

void Area_Calculator::setRadius(double a_radius)
{
    m_radius = a_radius;
}

double Area_Calculator::GetArea()
{
    return m_radius*m_radius*3.1415926;
}
```

→ 建構進行初始化

→ 公開方法探訪私有屬性

建構子：常用於初始化類別內的屬性資料或建立類別類所需之記憶體
解構子：常用於釋放類別內之物件或所使用的記憶體

嵌入式程式C/C++

```
#pragma once
```

```
class TArea
```

```
{
```

```
private:
```

```
    double m_width;
```

```
    double m_height;
```

```
    double m_baseSide;
```

```
    double m_radius;
```

```
public:
```

```
    void setTriangleParam(double a_base_side, double a_height);
```

```
    void setCircleParam(double a_radius);
```

```
    void setRectangleParam(double a_width, double a_height);
```

```
    double getTriangleArea();
```

```
    double getCircleArea();
```

```
    double getRectangleArea();
```

```
};
```

d.h

Example 3

```
#include <stdio.h>
```

```
#include "d.h"
```

```
int main()
```

```
{
```

```
    TArea *calculator = new TArea();
```

```
    calculator->setTriangleParam(4,4);
```

```
    double tri_area = calculator->getTriangleArea();
```

```
    calculator->setCircleParam(10);
```

```
    double circle_area = calculator->getCircleArea();
```

```
    calculator->setRectangleParam(6,8);
```

```
    double rect_area = calculator->getRectangleArea();
```

```
    printf("triangle area = %1.2f \n", tri_area);
```

```
    printf("circle area = %1.2f \n", circle_area);
```

```
    printf("rectangle area = %1.2f \n", rect_area);
```

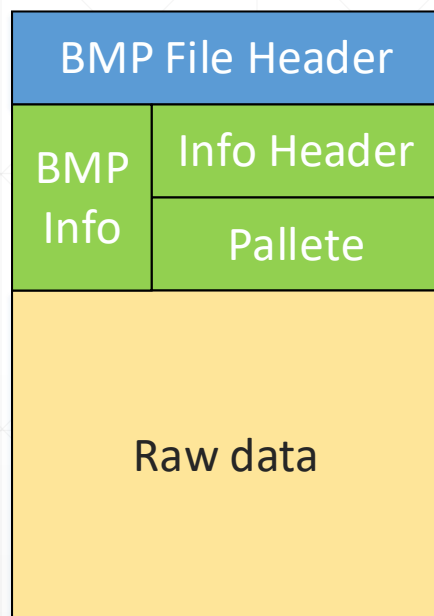
```
    delete calculator;
```

```
    return 0;
```

```
}
```

嵌入式程式C/C++

■ Bmp 影像格式



→ 檔頭資訊

→ 檔案詳細資訊

→ 調色盤資訊(如果需要的話) $N*4$ bytes ($N=2^8$)

→ 影像資料

8 bits gray image

14 bytes

40 bytes

$N*4$ bytes ($N=2^8$)

S bytes

File size = $14 + 40 + N*4 + S$

$S = \text{width} * \text{height} * \text{byte_per_pixel}$

Full color image (32 bits) : R、G、B、A channels

Color image (24 bits) : R、G、B channels

Gray image (8 bits) : 1 channels (gray level)

Gray level = $0.299*R + 0.587*G + 0.114*B$

24 bits color image

14 bytes

40 bytes

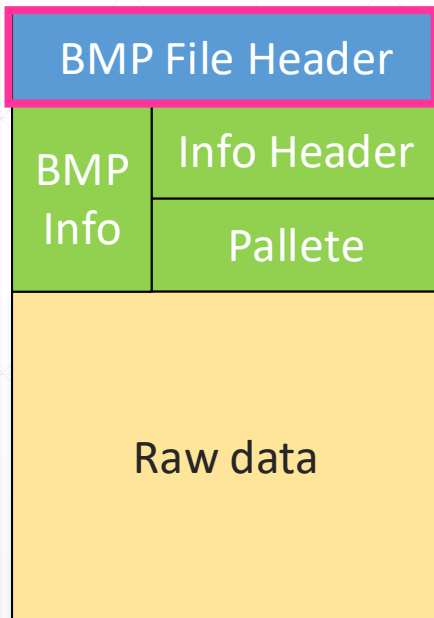
0 bytes

S bytes

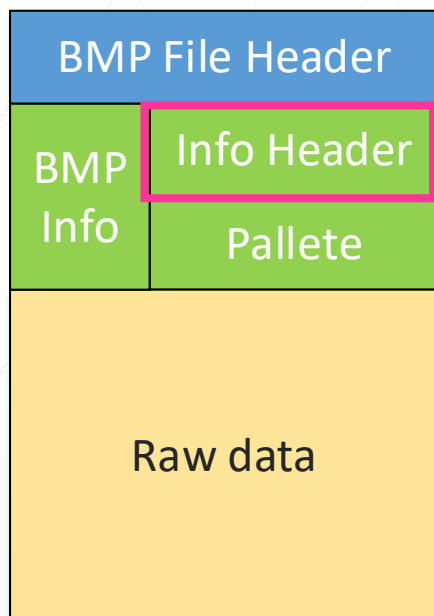
$14 + 40 + S$

BMP Info 也稱為 DIB資料結構

嵌入式程式C/C++



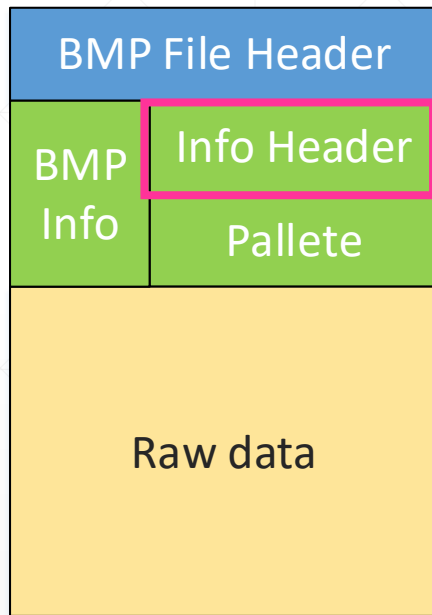
Name	Size (bytes)	Content
Identifier (ID)	2	'BM'
File Size	4	整個點陣圖檔案的大小 (單位 : byte)
Reserved	4	保留欄位
Bitmap Data Offset	4	點陣圖資料開始之前的偏移量 (單位 : byte)



Name	Size (bytes)	Content
Bitmap Header Size	4	Bitmap Info Header 的長度
Width	4	點陣圖的寬度，以像素 (pixel) 為單位
Height	4	點陣圖的高度，以像素 (pixel) 為單位
Planes	2	點陣圖的位元圖層數
Bits Per Pixel	2	每個像素的位元數 1：單色點陣圖 (使用 2 色調色盤) 4：4 位元點陣圖 (使用 16 色調色盤) 8：8 位元點陣圖 (使用 256 色調色盤) 16：16 位元高彩點陣圖 (不一定使用調色盤) 24：24 位元全彩點陣圖 (不使用調色盤) 32：32 位元全彩點陣圖 (不一定使用調色盤)
Compression	4	壓縮方式： 0：未壓縮 1： RLE 8-bit/pixel 2： RLE 4-bit/pixel 3： Bitfields
Bitmap Data Size	4	點陣圖資料的大小 (單位：byte)
H-Resolution	4	水平解析度 (單位：像素/公尺)
V-Resolution	4	垂直解析度 (單位：像素/公尺)
Used Colors	4	點陣圖使用的調色盤顏色數
Important Colors	4	重要的顏色數

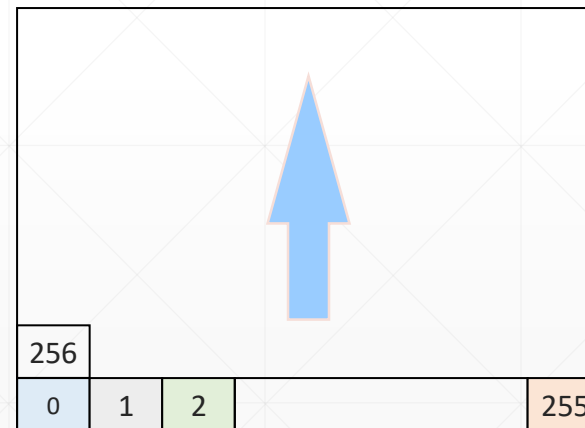
嵌入式程式C/C++

Name	Size (bytes)	Content
Height	4	點陣圖的高度，以像素 (pixel) 為單位

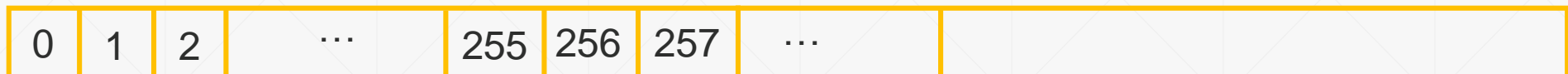


高度為正值，掃描方向由下而上。

高度為負值，掃描方向由上而下，亦代表此點陣圖不能被壓縮 (Compression = 0)

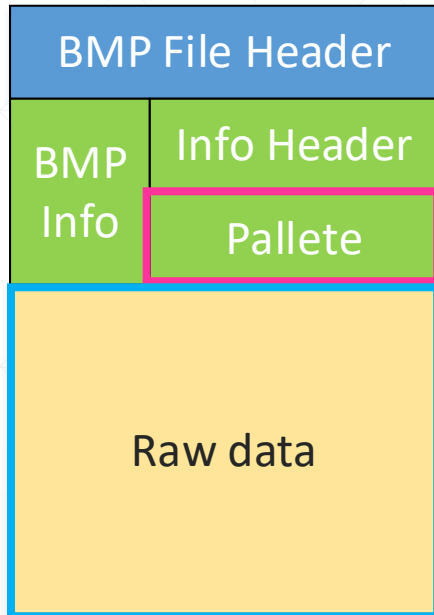


pixel



Raw data

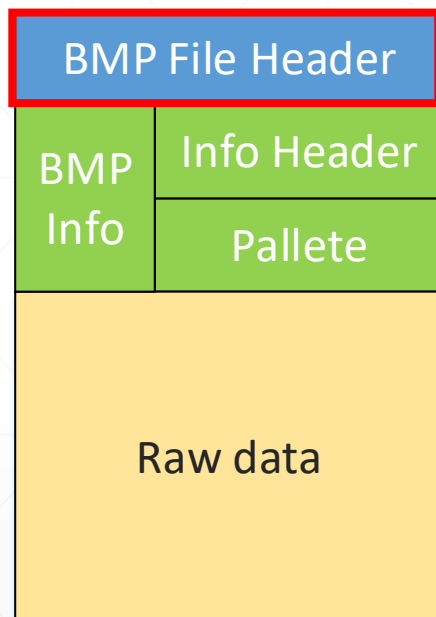
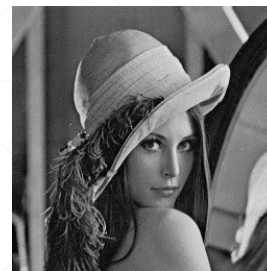
嵌入式程式C/C++



Name	Size (bytes)	Content
Palette	N*4	調色盤資料。 每個索引值指定一種顏色：0x00RRGGBB 其中最高位元組保留為零

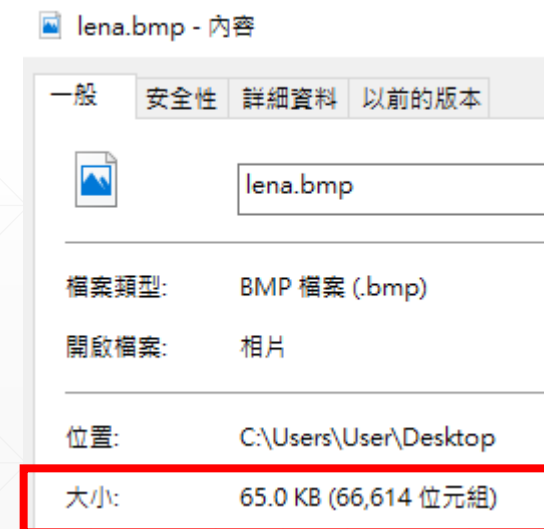
Name	Size (bytes)	Content
Bitmap Data		點陣圖資料

嵌入式程式C/C++



Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	42	4d	36	04	01	00	00	00	00	00	36	04	00	00	28	00	BM6.....6... (.
00000010	00	00	00	01	00	00	00	01	00	00	01	00	08	00	00	00
00000020	00	00	00	00	00	00	13	0b	00	00	13	0b	00	00	00	01

Name	Size (bytes)	Content
Identifier (ID)	2	'BM'
File Size	4	整個點陣圖檔案的大小 (單位 : byte)
Reserved	4	保留欄位
Bitmap Data Offset	4	點陣圖資料開始之前的偏移量 (單位 : byte)



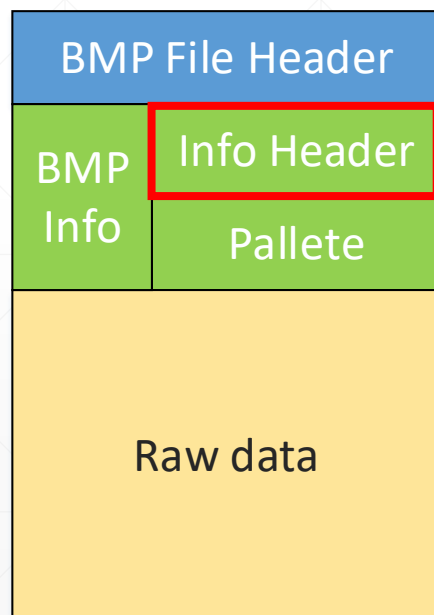
Big-Endian
Little-Endian

陣圖檔案的大小 : 00 01 04 36 (Hex) → 66,614 (Dec) Bytes

嵌入式程式C/C++



Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	D
00000000	42	4d	36	04	01	00	00	00	00	00	36	04	00	00	28	00	B
00000010	00	00	00	01	00	00	00	01	00	00	01	00	08	00	00	00	.
00000020	00	00	00	00	00	00	13	0b	00	00	13	0b	00	00	00	01	.
00000030	00	00	00	01	00	00	00	00	00	00	01	01	01	00	02	02	.



以列(row)為單位對齊

每一掃描列的長度必需是 4 bytes (32 bits) 的倍數

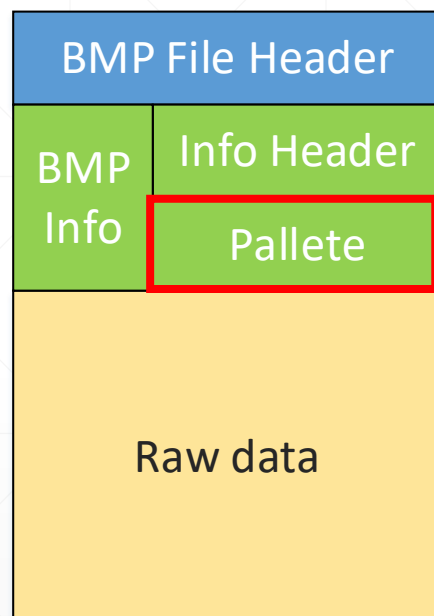
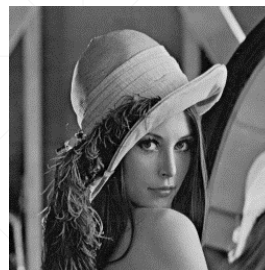
$$(\text{width} * \text{Bits Per Pixel} + 31) / 32 * 4$$

Row : 台灣 列 - 大陸 行
Column : 台灣 行 - 大陸 列

Big-Endian
Little-Endian

Name	Size (bytes)	Content	
		Hex	Dec
Bitmap Header Size	4	00000028	40
Width	4	00000100	256
Height	4	00000100	256
Planes	2	0001	8
Bits Per Pixel	2	0008	8
Compression	4	00000000	0
Bitmap Data Size	4	00000000	0
H-Resolution	4	00000b13	2834
V-Resolution	4	00000b13	2834
Used Colors	4	00000100	256
Important Colors	4	00000100	256

嵌入式程式C/C++



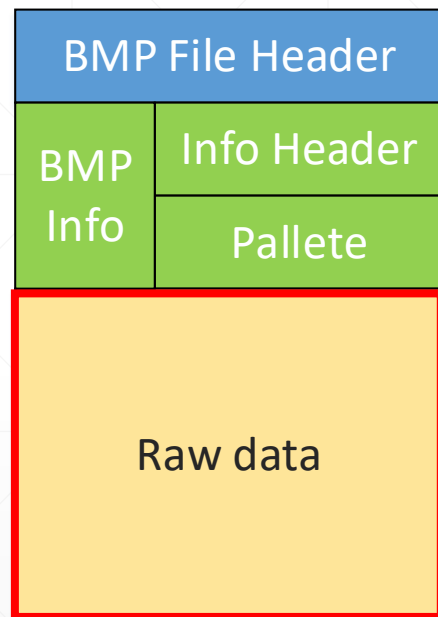
Big-Endian
Little-Endian

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	42	4d	36	04	01	00	00	00	00	00	36	04	00	00	28	00	BM6.....6... (.
00000010	00	00	00	01	00	00	00	01	00	00	01	00	08	00	00	00
00000020	00	00	00	00	00	00	13	0b	00	00	13	0b	00	00	00	01
00000030	00	00	00	01	00	00	00	00	00	01	01	01	00	02	02	
00000040	02	00	03	03	03	00	04	04	04	00	05	05	05	00	06	06
00000050	06	00	07	07	07	00	08	08	08	00	09	09	09	00	0a	0a
00000060	0a	00	0b	0b	0b	00	0c	0c	0c	00	0d	0d	0d	00	0e	0e
00000070	0e	00	0f	0f	0f	00	10	10	10	00	11	11	11	00	12	12
00000080	12	00	13	13	13	00	14	14	14	00	15	15	15	00	16	16
00000090	16	00	17	17	17	00	18	18	18	00	19	19	19	00	1a	1a
000000a0	1a	00	1b	1b	1b	00	1c	1c	1c	00	1d	1d	1d	00	1e	1e
000000b0	1e	00	1f	1f	1f	00	20	20	20	00	21	21	21	00	22	22
000000c0	22	00	23	23	23	00	24	24	24	00	25	25	25	00	26	26	".##+
000000d0	26	00	27	27	27	00	28	28	28	00	29	29	29	00	2a	2a	&.'''
000000e0	2a	00	2b	2b	2b	00	2c	2c	2c	00	2d	2d	2d	00	2e	2e	*.++-
000000f0	2e	00	2f	2f	2f	f0	30	30	30	00	31	31	31	00	32	32	..//,
00000100	32	00	33	33	33	00	34	34	34	00	35	35	35	00	36	36	2.33
00000110	36	00	37	37	37	00	38	38	38	00	39	39	39	00	3a	3a	6.77

Pallette : 第 55 bytes 到 第 1078 bytes = 1024 bytes

```
typedef struct tagRGBQUAD {
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

4 bytes * 256 gray level
= 1024 bytes



Big-Endian
Little-Endian

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000420	c3	86	c3	86	c3	86	00	c3	87	c3	87	c3	87	00	c3	88
00000430	c3	88	c3	88	00	c3	89	c3	89	c3	89	00	c3	8a	c3	8a
00000440	c3	8a	00	c3	8b	c3	8b	c3	8b	00	c3	8c	c3	8c	c3	8c
00000450	00	c3	8d	c3	8d	c3	8d	00	c3	8e	c3	8e	c3	8e	00	c3
00000460	8f	c3	8f	c3	8f	00	c3	90	c3	90	c3	90	00	c3	91	c3
00000470	91	c3	91	00	c3	92	c3	92	c3	92	00	c3	93	c3	93	c3
00000480	93	00	c3	94	c3	94	c3	94	00	c3	95	c3	95	c3	95	00
00000490	c3	96	c3	96	c3	96	00	c3	97	c3	97	c3	97	00	c3	98
000004a0	c3	98	c3	98	00	c3	99	c3	99	c3	99	00	c3	9a	c3	9a
000004b0	c3	9a	00	c3	9b	c3	9b	c3	9b	00	c3	9c	c3	9c	c3	9c
000004c0	00	c3	9d	c3	9d	c3	9d	00	c3	9e	c3	9e	c3	9e	00	c3
000004d0	9f	c3	9f	c3	9f	00	c3	a0	c3	a0	c3	a0	00	c3	a1	c3
000004e0	a1	c3	a1	00	c3	a2	c3	a2	c3	a2	00	c3	a3	c3	a3	c3
000004f0	a3	00	c3	a4	c3	a4	c3	a4	00	c3	a5	c3	a5	c3	a5	00
00000500	c3	a6	c3	a6	c3	a6	00	c3	a7	c3	a7	c3	a7	00	c3	a8
00000510	c3	a8	c3	a8	00	c3	a9	c3	a9	c3	a9	00	c3	aa	c3	aa
00000520	c3	aa	00	c3	ab	c3	ab	c3	ab	00	c3	ac	c3	ac	c3	ac
00000530	00	c3	ad	c3	ad	c3	ad	00	c3	ae	c3	ae	c3	ae	00	c3
00000540	af	c3	af	00	c3	ae	c3	ae	c3	ae	00	c3	af	c3	af	c3
00000550	b1	c3	b1	00	c3	af	c3	af	c3	af	00	c3	b0	c3	b0	c3
00000560	b3	00	c3	b4	c3	b4	c3	b4	00	c3	b5	c3	b5	c3	b5	00
00000570	c3	b6	c3	b6	c3	b6	00	c3	b7	c3	b7	c3	b7	00	c3	b8
00000580	c3	b8	c3	b8	00	c3	b9	c3	b9	c3	b9	00	c3	ba	c3	ba
00000590	c3	ba	00	c3	bb	c3	bb	c3	bb	00	c3	bc	c3	bc	c3	bc
000005a0	00	c3	bd	c3	bd	c3	bd	00	c3	be	c3	be	c3	be	00	c3
000005b0	bf	c3	bf	c3	bf	00	15	1e	19	1b	1f	17	1b	48	c2	99
000005c0	c2	bc	c2	be	c2	be	c2	bd	c2	ba	c2	ab	7a	53	60	6d
000005d0	c2	83	c2	8c	c2	95	c2	9d	c2	a2	c2	a1	c2	a1	c2	a3
000005e0	c2	a3	c2	9d	c2	95	4a	46	26	25	16	10	0e	22	29	0d
000005f0	28	12	30	32	22	2a	25	2d	34	25	1c	40	3b	c2	88	33
00000600	1c	2f	11	20	47	5e	12	28	34	2b	46	6e	31	21	78	45
00000610	11	1b	13	19	29	39	2d	21	18	1a	4c	2d	19	4c	37	51
00000620	19	2f	30	33	3a	38	3f	44	47	49	4c	51	53	57	56	57
00000630	59	57	5a	61	62	5a	5d	5c	5b	5d	5b	60	62	60	64	60

Raw data size :
每個row的資料量 * 行數(高度)
= ((256*8+31) / 32)*4 * 256
= 65536 bytes

每一掃描列的長度必需是
4 bytes (32 bits) 的倍數

Ex:

wid = 64

64 * 8bits = 512 bits

512 / 32 = 16 OK

所以寬度64pixel的影像，

每個row的資料量為：512 bits = 64 bytes

wid = 65

65 * 8bits = 520 bits

520 / 32 = 16.25 NG → 應為17

(520 + 31) / 32 = 17.21875 → 取實數

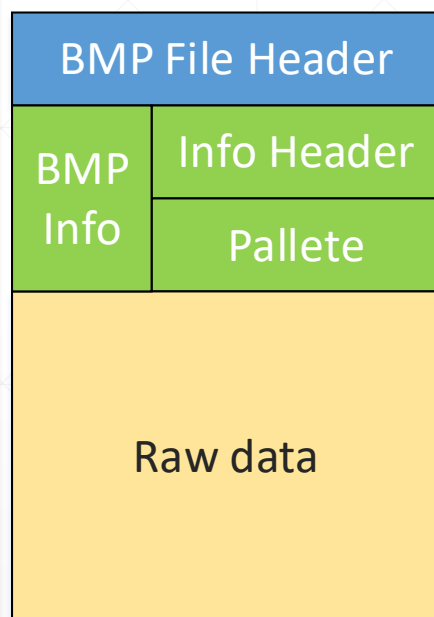
17 * 4 = 68 bytes

所以寬度65pixel的影像，

每個row的資料量為：68 bytes。

嵌入式程式C/C++

- 自建影像讀取類別
 - BMP檔案讀、存檔



```
typedef struct tagBITMAPFILEHEADER {  
    WORD  bfType;  
    DWORD bfSize;  
    WORD  bfReserved1;  
    WORD  bfReserved2;  
    DWORD bfOffBits;  
} BITMAPFILEHEADER, *LPBITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

```
typedef struct tagBITMAPINFO {  
    BITMAPINFOHEADER bmiHeader;  
    RGBQUAD          bmiColors[1];  
} BITMAPINFO, *LPBITMAPINFO, *PBITMAPINFO;
```

```
typedef struct tagBITMAPINFOHEADER {  
    DWORD biSize;  
    LONG  biWidth;  
    LONG  biHeight;  
    WORD  biPlanes;  
    WORD  biBitCount;  
    DWORD biCompression;  
    DWORD biSizeImage;  
    LONG  biXPelsPerMeter;  
    LONG  biYPelsPerMeter;  
    DWORD biClrUsed;  
    DWORD biClrImportant;  
} BITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

```
typedef struct tagRGBQUAD {  
    BYTE rgbBlue;  
    BYTE rgbGreen;  
    BYTE rgbRed;  
    BYTE rgbReserved;  
} RGBQUAD;
```

學習使用類別進行變數與記憶體管理
學習建立常用模組
學習如何隱匿程式碼

...

BYTE *rawdata

嵌入式程式C/C++



檔案路徑



MyBMPClass

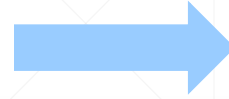
```
void  
usage (char *name)  
{  
    printf ("usage:\n");  
    printf ("%s -s [-c file",  
            name);  
    #ifdef LOG1  
        printf ("[-q] [-d] ");  
    #endif  
    printf ("[-u what] [-r]  
            [-u file (type)]");  
}
```

.C

```
void  
usage (char *name)  
{  
    printf ("usage:\n");  
    printf ("%s -s [-c file",  
            name);  
    #ifdef LOG1  
        printf ("[-q] [-d] ");  
    #endif  
    printf ("[-u what] [-r]  
            [-u file (type)]");  
}
```

.H

公開方法




```
typedef struct tagBITMAPFILEHEADER {
    WORD  bfType;
    DWORD bfSize;
    WORD  bfReserved1;
    WORD  bfReserved2;
    DWORD bfOffBits;
} BITMAPFILEHEADER, *LPBITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

MyBMPClass



```
typedef struct tagBITMAPINFOHEADER {
    DWORD biSize;
    LONG  biWidth;
    LONG  biHeight;
    WORD  biPlanes;
    WORD  biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG  biXPelsPerMeter;
    LONG  biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

```
typedef struct tagRGBQUAD {
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

BYTE *rawdata

```
class MyImageClass
{
private:
    //Data
    //BMP Info
    int wid;
    int hei;
    //...

    BYTE *rawdata;

public:
    MyImageClass();
    ~MyImageClass();

public:
    bool LoadBMPfile(char *file);
    //...

    int GetImageWidth();
    int GetImageHeight();
    //...
};
```



思維：應用反推

嵌入式程式C/C++

Windows CImage 類別

名稱	描述
CImage::AlphaBlend	顯示具有透明或半透明的像素的點陣圖。
CImage::Attach	附加至 HBITMAPCImage 物件。 可以搭配非 DIB 區段點陣圖或 DIB 區段點陣圖。
CImage::BitBlt	來源裝置內容的點陣圖複製到這個目前的裝置內容。
CImage::Create	建立 DIB 區段點陣圖，並將它附加至先前建構 CImage 物件。
CImage::CreateEx	建立 DIB 區段點陣圖（與其他參數），並將它附加至先前建構 CImage 物件。
CImage::Destroy	卸離點陣圖 CImage 物件，並終結點陣圖。
CImage::Detach	卸離點陣圖 CImage 物件。
CImage::Draw	來源矩形的點陣圖複製到目的地矩形。 Draw 會自動縮放或壓縮點陣圖以符合目的地矩形的維度，如有必要，並處理 alpha 透明混色和透明的色彩。
CImage::GetBits	擷取點陣圖的實際像素值的指標。
CImage::GetBPP	擷取每個像素的位元。
CImage::GetColorTable	從範圍的色彩表中的項目擷取紅、綠、藍 (RGB) 色彩值。
CImage::GetDC	擷取到其中的目前點陣圖已選取的裝置內容。
CImage::GetExporterFilterString	尋找可用映像格式和其描述。
CImage::GetHeight	擷取目前的映像，單位為像素的高度。
CImage::GetImporterFilterString	尋找可用映像格式和其描述。

CImage::GetMaxColorTableEntries	擷取色彩表中的項目的數目上限。
CImage::GetPitch	擷取目前的映像，以位元組為單位的字幅。
CImage::GetPixel	擷取所指定的像素的色彩x並y。
CImage::GetPixelAddress	擷取指定的像素的位址。
CImage::GetTransparentColor	擷取的透明色彩的色彩表中的位置。
CImage::GetWidth	擷取目前的影像像素為單位的寬度。
CImage::IsDIBSection	判斷附加的點陣圖為之 DIB 區段。
CImage::IsIndexed	表示點陣圖的色彩會對應到索引的調色盤。
CImage::IsNull	表示是否目前已載入的來源點陣圖。
CImage::IsTransparencySupported	指出應用程式是否支援透明的點陣圖。
CImage::Load	從指定的檔案載入影像。
CImage::LoadFromResource	從指定的資源載入映像。
CImage::MaskBlt	結合使用指定的遮罩和點陣作業的來源和目的地點陣圖的色彩資料。
CImage::PlgBlt	執行從來源裝置內容中的矩形的位元區塊傳輸到目的地裝置內容中的平行四邊形。
CImage::ReleaseDC	釋放與已擷取的裝置內容 CImage::GetDC 。
CImage::ReleaseGDIPlus	釋放 GDI + 所使用的資源。 必須由全域可用的資源呼叫 CImage 物件。
CImage::Save	將影像儲存為指定的類型。 Save 無法指定映像的選項。
CImage::SetColorTable	將紅色、綠色、藍色 RGB) 色彩的色彩表之 DIB 區段中的項目範圍中的值。
CImage::SetPixel	在指定的座標，以指定的色彩設定像素。
CImage::SetPixelIndexed	色彩調色盤的指定索引處的指定座標上設定之像素。
CImage::SetPixelRGB	在指定的座標指定的紅、綠、藍 (RGB) 值會設定像素。
CImage::SetTransparentColor	會設定為透明處理色彩的索引。 只有一種色彩調色盤中的可以是透明的。
CImage::StretchBlt	來源矩形的點陣圖複製到目的地矩形，延伸或壓縮點陣圖以符合目的地矩形的維度，如有必要。
CImage::TransparentBlt	來源裝置內容以透明色彩的點陣圖複製到這個目前的裝置內容。

■ Qt QImage 類別

- >
- >
- >
- > enum **Format**
- > enum **InvertMode**
- > enum **PaintDeviceMetric**
- > typedef **QtGadgetHelper**
- > **QImage()**
- > **QImage**(const QSize &, QImage::Format)
- > **QImage**(int , int , QImage::Format)
- > **QImage**(uchar *, int , int , QImage::Format ,
QImageCleanupFunction , void *)
- > **QImage**(const uchar *, int , int , QImage::Format ,
QImageCleanupFunction , void *)
- > **QImage**(uchar *, int , int , int , QImage::Format ,
QImageCleanupFunction , void *)
- > **QImage**(const uchar *, int , int , int , QImage::Format ,
QImageCleanupFunction , void *)
- > **QImage**(const char *const [])
- > **QImage**(const QString &, const char *)
- > **QImage**(const QImage &)
- > **QImage**(QImage &&)
- > **~QImage()**
- > **allGray()** const : bool
- > **bitPlaneCount()** const : int
- > **bits()** : uchar *
- > **bits()** const : const uchar *
- > **bytesPerLine()** const : int

- > **mirrored**(bool , bool) : QImage &&
- > **offset()** const : QPoint
- > **paintEngine()** const : QPaintEngine *
- > **painters** : ushort
- > **paintingActive()** const : bool
- > **physicalDpiX()** const : int
- > **physicalDpiY()** const : int
- > **pixel**(const QPoint &) const : QRgb
- > **pixel**(int , int) const : QRgb
- > **pixelColor**(const QPoint &) const : QColor
- > **pixelColor**(int , int) const : QColor
- > **pixelFormat()** const : QPixelFormat
- > **pixelIndex**(const QPoint &) const : int
- > **pixelIndex**(int , int) const : int
- > **qt_check_for_QGADGET_macro()**
- > **rect()** const : QRect
- > **reinterpretAsFormat**(QImage::Format) : bool
- > **rgbSwapped()** const : QImage
- > **rgbSwapped()** : QImage &&
- > **save**(const QString &, const char *, int) const : bool
- > **save**(QIODevice *, const char *, int) const : bool
- > **scaled**(const QSize &, Qt::AspectRatioMode ,
Qt::TransformationMode) const : QImage
- > **scaled**(int , int , Qt::AspectRatioMode ,
Qt::TransformationMode) const : QImage
- > **scaledToHeight**(int , Qt::TransformationMode) const :
QImage

■ Qt QImage 類別

- › `cacheKey()` const : qint64
- › `color(int)` const : QRgb
- › `colorCount()` const : int
- › `colorTable()` const : QVector<QRgb>
- › `constBits()` const : const uchar *
- › `constScanLine(int)` const : const uchar *
- › `convertToFormat(QImage::Format, Qt::ImageConversionFlags)` const : QImage
- › `convertToFormat(QImage::Format, Qt::ImageConversionFlags)` : QImage
- › `convertToFormat(QImage::Format, const QVector<QRgb> &, Qt::ImageConversionFlags)` const : QImage
- › `copy(const QRect &)` const : QImage
- › `copy(int, int, int, int)` const : QImage
- › `createAlphaMask(Qt::ImageConversionFlags)` const : QImage
- › `createHeuristicMask(bool)` const : QImage
- › `createMaskFromColor(QRgb, Qt::MaskMode)` const : QImage
- › `depth()` const : int
- › `devicePixelRatio()` const : qreal
- › `devicePixelRatioF()` const : qreal
- › `dotsPerMeterX()` const : int
- › `dotsPerMeterY()` const : int
- › `fill(uint)`
- › `fill(const QColor &)`
- › `fill(Qt::GlobalColor)`
- › `format()` const : QImage::Format
- › `fromData(const uchar *, int, const char *)` : QImage
- › `scaledToWidth(int, Qt::TransformationMode)` const : QImage
- › `scanLine(int)` : uchar *
- › `scanLine(int)` const : const uchar *
- › `setColor(int, QRgb)`
- › `setColorCount(int)`
- › `setColorTable(const QVector<QRgb>)`
- › `setDevicePixelRatio(qreal)`
- › `setDotsPerMeterX(int)`
- › `setDotsPerMeterY(int)`
- › `setOffset(const QPoint &)`
- › `setPixel(const QPoint &, uint)`
- › `setPixel(int, int, uint)`
- › `setPixelColor(const QPoint &, const QColor &)`
- › `setPixelColor(int, int, const QColor &)`
- › `setText(const QString &, const QString &)`
- › `size()` const : QSize
- › `sizeInBytes()` const : qsize_t
- › `smoothScaled(int, int)` const : QImage
- › `staticMetaObject` : const QMetaObject
- › `swap(QImage &)`
- › `text(const QString &)` const : QString
- › `textKeys()` const : QStringList
- › `toCGLImage()` const : CGLImageRef
- › `toImageFormat(QPixelFormat)` : QImage::Format
- › `toPixelFormat(QImage::Format)` : QPixelFormat

嵌入式程式C/C++

▪ Qt QImage 類別

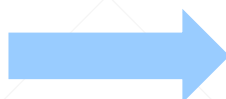
- › **fromData**(const QByteArray &, const char *) : QImage
- › **hasAlphaChannel**() const : bool
- › **height**() const : int
- › **heightMM**() const : int
- › **invertPixels**(QImage::InvertMode)
- › **isGrayscale**() const : bool
- › **isNull**() const : bool
- › **load**(const QString &, const char *) : bool
- › **load**(QIODevice *, const char *) : bool
- › **loadFromData**(const uchar *, int, const char *) : bool
- › **loadFromData**(const QByteArray &, const char *) : bool
- › **logicalDpiX**() const : int
- › **logicalDpiY**() const : int
- › **metric**(QPaintDevice::PaintDeviceMetric) const : int
- › **mirrored**(bool, bool) const : QImage

- › **transformed**(const QMatrix &, Qt::TransformationMode) const : QImage
- › **transformed**(const QTransform &, Qt::TransformationMode) const : QImage
- › **trueMatrix**(const QMatrix &, int, int) : QMatrix
- › **trueMatrix**(const QTransform &, int, int) : QTransform
- › **valid**(const QPoint &) const : bool
- › **valid**(int, int) const : bool
- › **width**() const : int
- › **widthMM**() const : int
- › **operator QVariant**() const : QVariant
- › **operator!=**(const QImage &) const : bool
- › **operator=**(const QImage &) : QImage &
- › **operator=**(QImage &&) : QImage &
- › **operator==**(const QImage &) const : bool

嵌入式程式C/C++

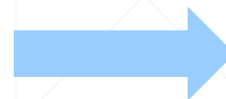


檔案路徑

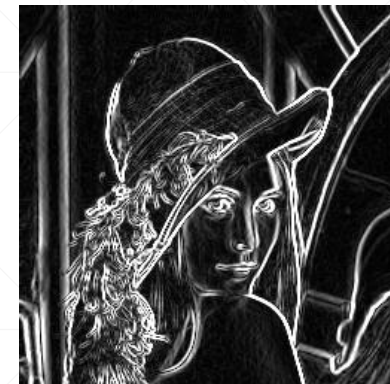


QImage
QPixmap
QBitmap
QPicture

公開方法



QtGUI 圖像顯示元件



演算法

Qt provides four classes for handling image data: [QImage](#), [QPixmap](#), [QBitmap](#) and [QPicture](#).

- [QImage](#) is designed and optimized for I/O, and for direct pixel access and manipulation,
- [QPixmap](#) is designed and optimized for showing images on screen.
- [QBitmap](#) is only a convenience class that inherits [QPixmap](#)
- [QPicture](#) class is a paint device that records and replays [QPainter](#) commands.

嵌入式程式C/C++



QtGUI 圖像顯示元件

QGraphicsView 元件

QGraphicsScene + QPixmap + QImage



嵌入式程式C/C++

- 請建立新專案，加入三個按鈕(Load Image、Load Image2、Threshold)、一個顯示元件(QGraphicsView)。

全域變數

```
QGraphicsScene *scene;  
QImage *g_img;  
  
MainWindow::MainWindow(QWidget *parent) :  
    QMainWindow(parent),  
    ui(new Ui::MainWindow)  
{  
    ui->setupUi(this);  
  
    scene = new QGraphicsScene();  
}  
  
MainWindow::~~MainWindow()  
{  
    delete ui;  
    delete scene;  
    delete g_img;  
}
```


嵌入式程式C/C++

- 進行讀取BMP檔與顯示 -- 按鈕一：Load Image

```
void MainWindow::on_pushButton_clicked()
{
    QString filename = "lena.bmp";
    QImage img = QImage(filename, "BMP"); 透過檔案進行QImage的建立，目前寫法需要圖檔位於專案路徑下

    ui->graphicsView->setFixedWidth(img.width());
    ui->graphicsView->setFixedHeight(img.height()); 設定GraphicsView的大小

    QPixmap pixmap = QPixmap::fromImage(img); 產生QPixmap

    scene->clear();
    scene->setSceneRect(0,0,img.width(),img.height()); 清除舊有資料並重新貼上pixmap影像資料
    scene->addPixmap(pixmap);

    ui->graphicsView->setScene(scene); 將scene顯示於QGraphicsview上
    ui->graphicsView->show();
}
```

QImage → QPixmap → QGraphicsScene → QGraphicsView

影像資料

資料轉換

顯示零件

介面顯示元件

資料



顯示

嵌入式程式C/C++

- 進行讀取BMP檔與顯示 -- 按鈕二：Load Image2
- 使用QFileDialog元件獲取檔案名稱與路徑

```
void MainWindow::on_pushButton_2_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, "Open image file", ".",
                                                    "Image files (*.bmp);;All Files (*.*)");
    if(fileName != "")
    {
        QByteArray filebyt = fileName.toLocal8Bit();
        char *filename = filebyt.data();

        g_img = new QImage(filename);

        ui->graphicsView->setFixedWidth(g_img->width());
        ui->graphicsView->setFixedHeight(g_img->height());

        QPixmap pixmap = QPixmap::fromImage(*g_img);

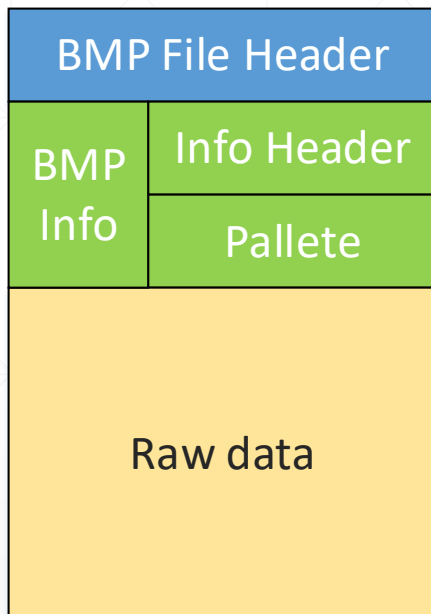
        scene->clear();
        scene->setSceneRect(0,0,g_img->width(),g_img->height());
        scene->addPixmap(pixmap);

        ui->graphicsView->setScene(scene);
        ui->graphicsView->show();
    }
}
```

嵌入式程式C/C++

- 進行影像二值化處理 → 從QImage獲取影像 Raw data 進行演算法

```
g_img = new QImage(filename);
```



```
uchar *QImage::bits()
```

Returns a pointer to the first pixel data. This is equivalent to `scanLine(0)`.

Note that `QImage` uses `implicit data sharing`. This function performs a deep copy of the shared pixel data, thus ensuring that this `QImage` is the only one using the current return value.

```
uchar *QImage::scanLine(int i)
```

Returns a pointer to the pixel data at the scanline with index `i`. The first scanline is at index 0.

The scanline data is aligned on a 32-bit boundary.

Warning: If you are accessing 32-bpp image data, cast the returned pointer to `QRgb*` (`QRgb` has a 32-bit size) and use it to read/write the pixel value. You cannot use the `uchar*` pointer directly, because the pixel format depends on the byte order on the underlying platform. Use `qRed()`, `qGreen()`, `qBlue()`, and `qAlpha()` to access the pixels.


```
void MainWindow::on_pushButton_3_clicked()
{
    int wid = g_img->width();
    int hei = g_img->height();

    QImage grayImg;

    QVector<QRgb> table;
    for(int i=0;i<256;i++)
        table.push_back(qRgb(i,i,i));

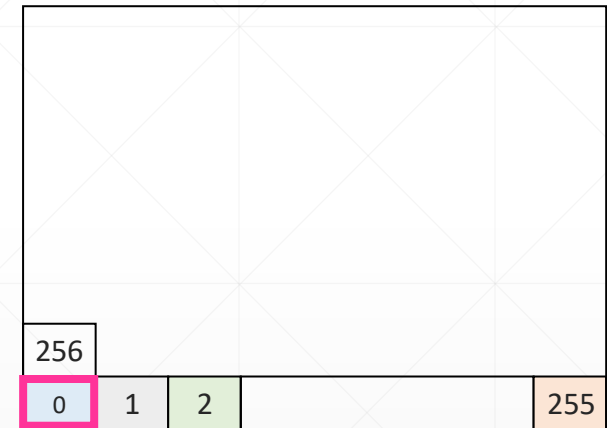
    grayImg = g_img->convertToFormat(QImage::Format_Grayscale8, table);

    uchar *data = grayImg.bits();
    for(int i = 0; i < wid*hei; i++)
    {
        int value = data[i];
        if(value > 128)
            data[i] = 255;
        else
            data[i] = 0;
    }


    QPixmap pixmap = QPixmap::fromImage(grayImg);

    scene->clear();
    scene->setSceneRect(0,0,grayImg.width(),grayImg.height());
    scene->addPixmap(pixmap);


    ui->graphicsView->setScene(scene);
    ui->graphicsView->show();
}
```



- 



機器視覺演算法


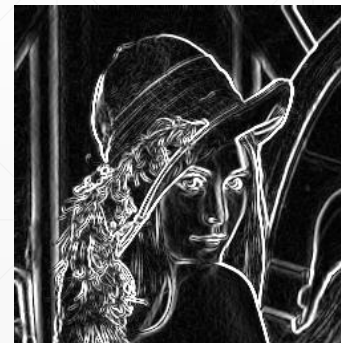


```
void usage(char *name)
{
    printf("usage: %s\\n",
name);
    printf("%s -a <C file>\\n",
name);
    if(argc < 3)
    {
        fprintf(stderr, "C: bad %d\\n", 1);
        return 1;
    }
    printf("C: %s\\n", argv[2]);
    if(!fopen(argv[2], "r")==f)
    {
        fprintf(stderr, "C: %s file doesn't exist\\n", argv[2]);
        return 1;
    }
}
```

.C

```
void usage(char *name)
{
    printf("usage: %s\\n",
name);
    printf("%s -a <C file>\\n",
name);
    if(argc < 3)
    {
        fprintf(stderr, "C: bad %d\\n", 1);
        return 1;
    }
    printf("C: %s\\n", argv[2]);
    if(!fopen(argv[2], "r")==f)
    {
        fprintf(stderr, "C: %s file doesn't exist\\n", argv[2]);
        return 1;
    }
}
```

.H



UI

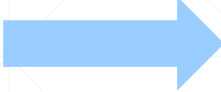
31

嵌入式程式C/C++

MyBMPClass



公開方法



```
void MainWindow::on_pushButton_3_clicked()
{
    int wid = g_img->width();
    int hei = g_img->height();

    QImage grayImg;

    QVector<QRgb> table;
    for(int i=0;i<256;i++)
        table.push_back(qRgb(i,i,i));

    grayImg = g_img->convertToFormat(QImage::Format_Grayscale8, table);

    uchar *data = grayImg.bits();
    for(int i = 0; i < wid*hei; i++)
    {
        int value = data[i];
        if(value > 128)
            data[i] = 255;
        else
            data[i] = 0;
    }

    QPixmap pixmap = QPixmap::fromImage(grayImg);

    scene->clear();
    scene->setSceneRect(0,0,grayImg.width(),grayImg.height());
    scene->addPixmap(pixmap);

    ui->graphicsView->setScene(scene);
    ui->graphicsView->show();
}
```