暑期課程
基本影像處理 day 2

指導教授：顏淑惠、林慧珍

HTTP://163.13.127.10

HTTP://PRIA.CS.TKU.EDU.TW

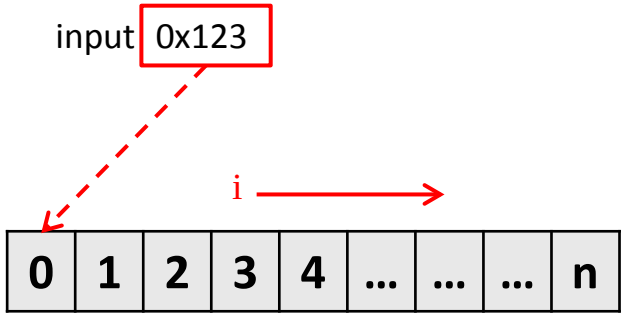
指導教授：凃瀞珽

HTTP://MAIL.TKU.EDU.TW/CTTU

Outline

- C++動態陣列宣告 : Image to Array
- 彩色影像(Color Image)的表示與存取
- HW2-1 : convert RGB to gray
- HW2-2 : Color Image->RGB planes (取出RGB三原色)
- Image Histogram(直方圖) and Binarize(二值化) 介紹
- HW2-3 : 二值化

Image to Array



- 一維動態陣列

```
//讀圖
Mat Input = imread("lena_pic24bit.bmp",-1);

int h = Input.rows; //影像高
int w = Input.cols; //影像寬
int length = h * w; //影像長度
//int length = 10;          //若不讀圖記得給陣列長度

//宣告 1D array 並配置
int *input = new int[length];
/*
宣告與配置分開
int *input;
input = new int[length];
*/

for(int i =0;i<length;i++){
    //...
}
//有宣告new就要有刪除delete
delete[] input;     // []表示是要刪除一塊配置給array的記憶體空間
```

int *input;

⇔ (int *) input;

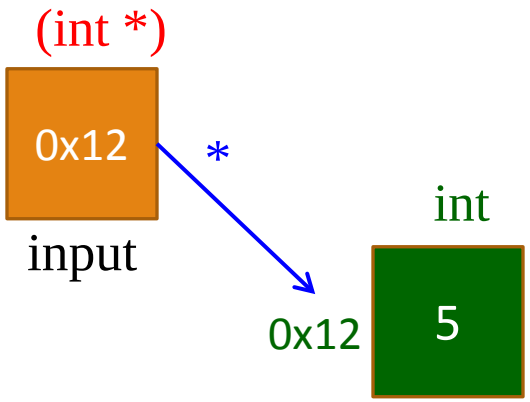
⇔ int* input;

例:

int *input;

input = new int;

* (宣告input是指標)



*input = 5;

* (指向input位址所指的位置並賦予值)

delete input; //delete 0x12

```
int *a;
a = new int;
*a = 5;

cout << a << endl;
cout << *a << endl;

delete a;
```

Image to Array

- delete
  - 假設input內存記憶體位址0x120

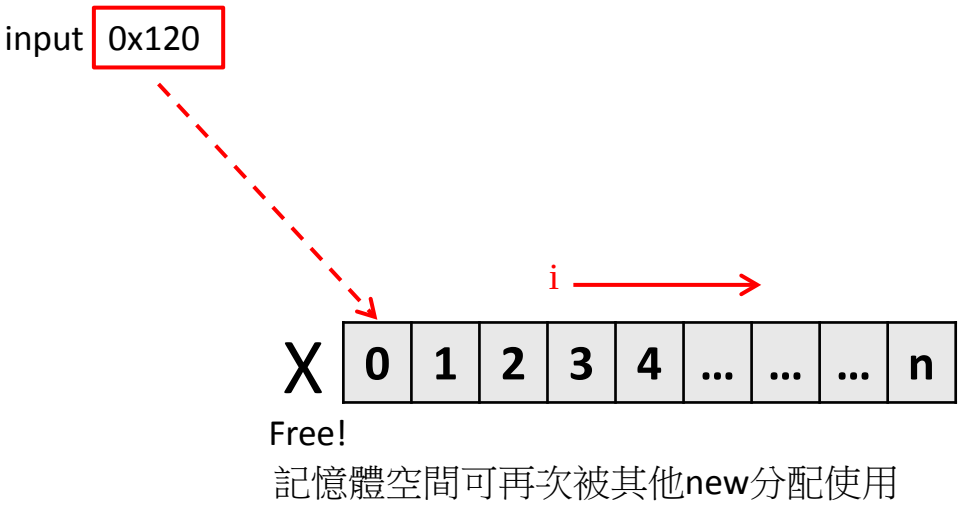


Image to 2D Array

- 二維動態陣列

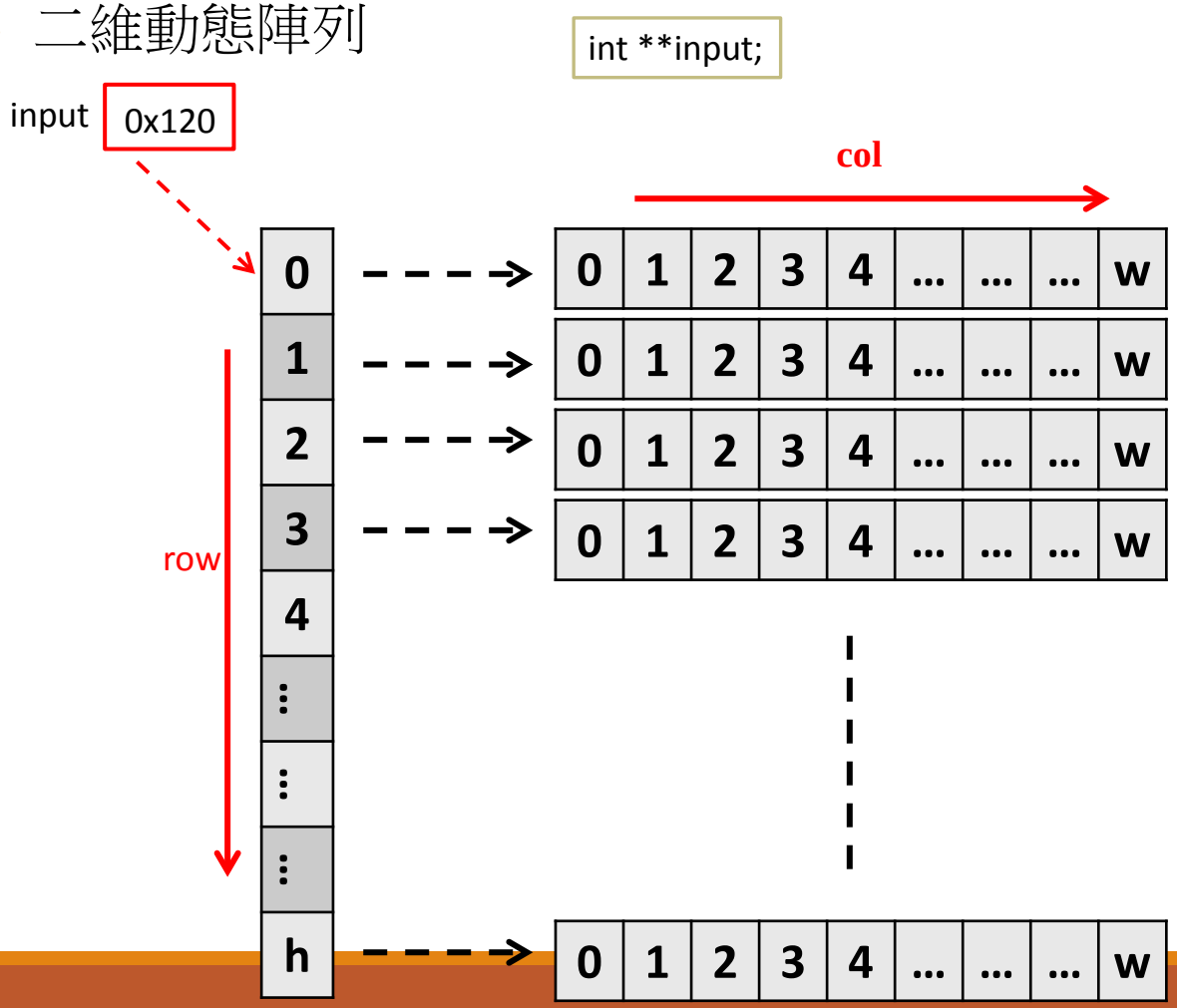


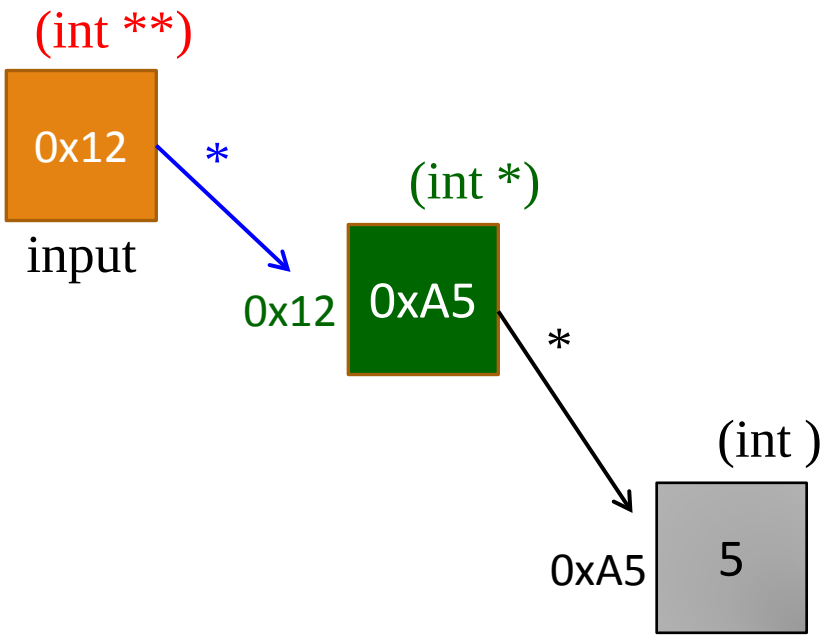
Image to 2D Array

```
//讀圖
Mat Input = imread("lena_pic24bit.bmp",-1);

int h = Input.rows;      //影像高
int w = Input.cols;      //影像寬

//宣告 2D array 並配置
int **input = new int*[h];       //配置第一維陣列,長度為hight
                                 //所配置的每一個int[h]所存都是一維指標int*
for(int row=0;row<h;row++)
{
    input[row] = new int[w];     //再將第一維的每格配置成一維陣列
                                 //及第二維陣列
}                                //所配置的每格input[row][col]都是int
                                 ex: Input[row][col] = gray;
//release memory
for(int row=0;row<h;row++)
{
    delete[] input[row];     //從內層往外層刪,順序與宣告時相反
}
delete[] input;      //刪除外層
```

int **input;

⇔ (int **) input;

⇔ int** input;



例:

int **input;

input = new int*;

*input = new int;

* (宣告input是指標)

**input = 5;

⇔ *(*input) = 5;

* (指向input位址所指的位置並賦予值)

delete *input; //delete 0xA5

delete input; //delete 0x12

```
int **b;
b = new int*;
*b = new int;
**b = 3;
cout << b << endl;
cout << *b << endl;
cout << **b << endl;

delete *b;
delete b;
```

Image to 2D Array

```
//讀圖
Mat Input = imread("lena_pic24bit.bmp",-1);

int h = Input.rows;      //影像高
int w = Input.cols;      //影像寬

//宣告 2D array 並配置
int **input = new int*[h];       //配置第一維陣列,長度為hight
                                 //所配置的每一個int[h]所存都是一維指標int*
for(int row=0;row<h;row++)
{
    input[row] = new int[w];     //再將第一維的每格配置成一維陣列
                                 //及第二維陣列
}                                //所配置的每格input[row][col]都是int
                                 ex: Input[row][col] = gray;
//release memory
for(int row=0;row<h;row++)
{
    delete[] input[row];     //從內層往外層刪,順序與宣告時相反
}
delete[] input;      //刪除外層
```

Image to 2D vector

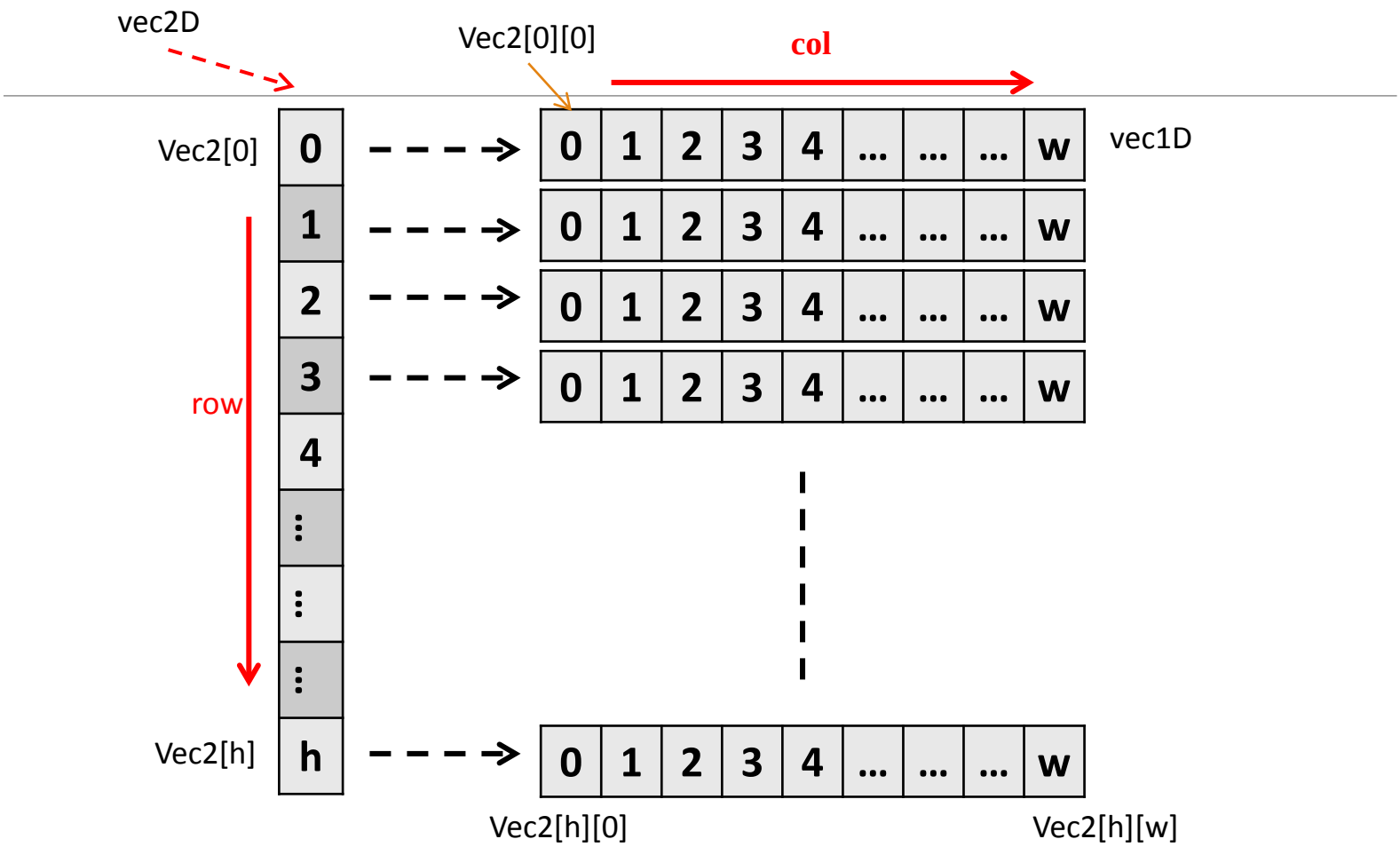


Image to 2D vector

```
//vector<int> vec1D(3);                    //1D vector, size = 3
//vector< vector<int> > vec2D(5, vec1D);   //2D vector, size = 5x3

int height = input.rows;
int width = input.cols;

vector<int> vec1D(width);                  //1D vector, size = width
vector<vector<int>> vec2D(height, vec1D);  //2D vector, size = height

vec2D[row][col] = 255;                     //for loop

cout << vec2D.size() << endl;              //height
cout << vec2D[0].size() << endl;           //width

cout << vec1D.size() << endl;
cout << vec1D.capacity() << endl;
cout << vec1D.max_size() << endl;
```

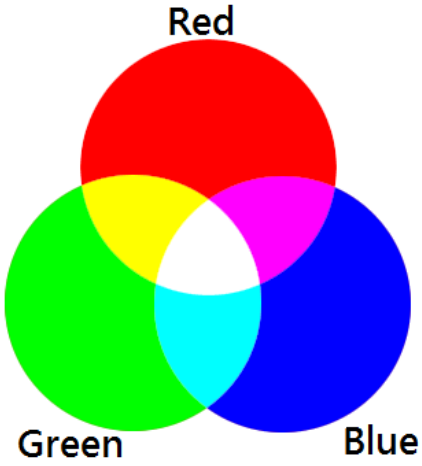
與array建立剛好相反,
先建立影像寬的1D vector,
再透過1D vector建立2D vector

Image to 2D vector (附件)

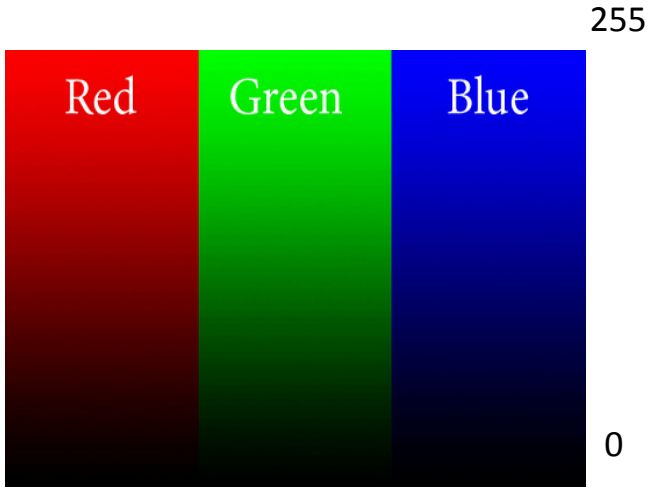
vector v(100);

檢查是否為空：v.empty();
直接將**v**清空：v.clear();
檢查目前大小：v.size();
檢查最大容量：v.maxsize();
檢查目前容量：v.captacity();
清除所有內容：v.clear();
清除指定元素：v.erase(v.begin()); v.erase(v.end()-1);
讀取並刪除最後一元素：v.pop_back(v.size()-1);
重新設定大小：v.resize(5);

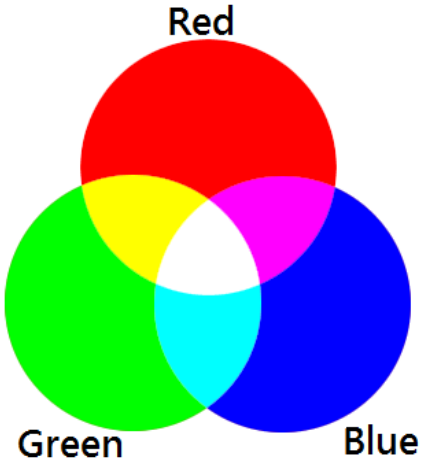
彩色影像 (Color Image)



- 3 channels (R, G, B)
  - 將色光三原色做量化
  - 1 pixel 為 3 bytes 大小
  - 即每 1 pixel 可存的值為 $0 \sim 256^3 - 1$
    => 0~0xFFFFFF (pixel values) => 共 256^3 種顏色
- 意即將三原色個別量化成0~255之間
  - 越接近 0 代表亮度越低
  - 越接近 255 代表亮度越高



彩色影像 (Color Image)



彩色影像 (Color Image)

- 3 channels (R, G, B)
- 小畫家 => 調色盤
- For example

	(R, G, B)	Pixel值
	(255, 0, 0)	0xFF0000
	(0, 255, 0)	0x00FF00
	(0, 0, 255)	0x0000FF
	(0, 0, 0)	0x000000
	(0, 255, 255)	0x00FFFF
	(255, 255, 0)	0xFFFF00
	(172, 0, 0)	0xAC0000
	(255, 192, 0)	0xFFC000

彩色影像 (Color Image)

pixel values of 24-bit image in memory

- 記憶體中讀取 channel 順序為BGR
  - 若每次只取1 channel , 則對於寬度w的圖片 , 每列所取的次數為w*3
  - e.g. 2x3 image

For col = 1 to 3*3

For row = 1 to 2

Pixel 1			Pixel 2			Pixel 3		
B	G	R	B	G	R	B	G	R
Pixel 4			Pixel 5			Pixel 6		
B	G	R	B	G	R	B	G	R

彩色影像 (Color Image)　轉灰階範例

```
Mat img = imread("lena_24bit.bmp", 1);
Mat Output( img.rows, img.cols, CV_8UC3, Scalar(0,0,0) );  //根據Mat建構元宣告

int h = img.rows;
int w = img.cols;

for (int row = 0 ; row<h ; row++) {
    for (int col = 0 ; col<w ; col++) {  一次取3 channel故仍為w
                                                                    //取出BGR計算灰階值
        uchar val = ( img.at<Vec3b>(row,col)[0] + img.at<Vec3b>(row,col)[1] + img.at<Vec3b>(row,col)[2] ) / 3;

        Output.at<Vec3b>(row,col)[0] = val;    //1st byte at (row, col), Blue byte
        Output.at<Vec3b>(row,col)[1] = val;    //2nd byte at (row, col), Green byte  //將灰階值存回Mat
        Output.at<Vec3b>(row,col)[2] = val;    //3rd byte at (row, col), Red byte
    }              <Vec3b> 意即 (row, col)位置為3 byte vector
}

namedWindow("out",0);
imshow("out", Output);
```

Homework #2.1

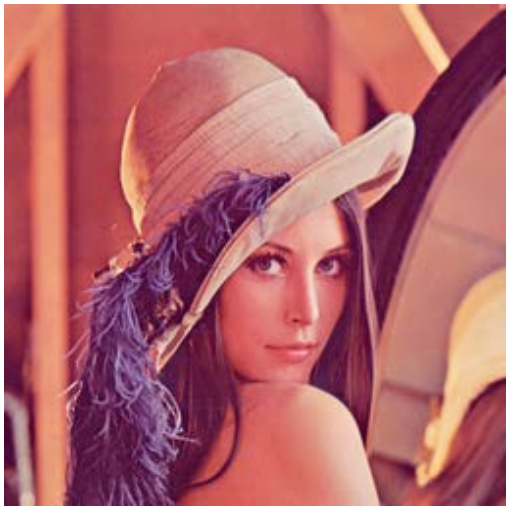
✓convert RGB to Gray (請用公式解：gray = R*0.299 + G*0.587 + B*0.114)

使用Mat讀圖後，存到動態二維陣列中做灰階運算，最後再存回Mat做Output

即：讀圖至Mat => Mat轉2D Array => 取出R, G, B Array => Gray Array => Mat

測試圖：

1. Lena
2. 隨便找張圖測試



Homework #2.2

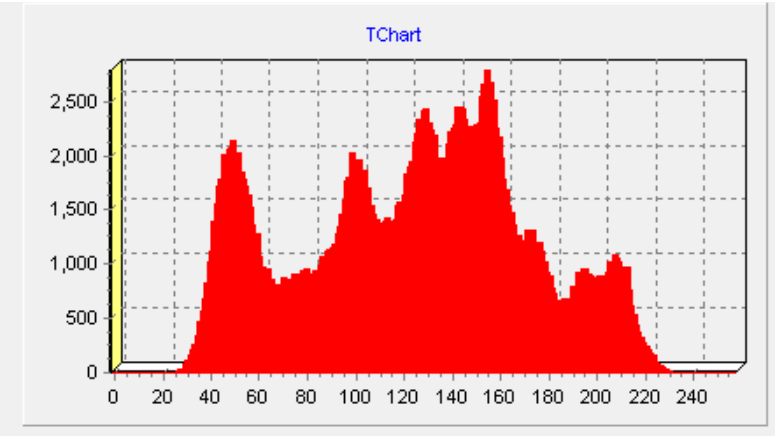
✓RGB to R,G,B planes

只保留目標channel, 其他channel設為0

Image2,3,4 for showing R,G,B planes

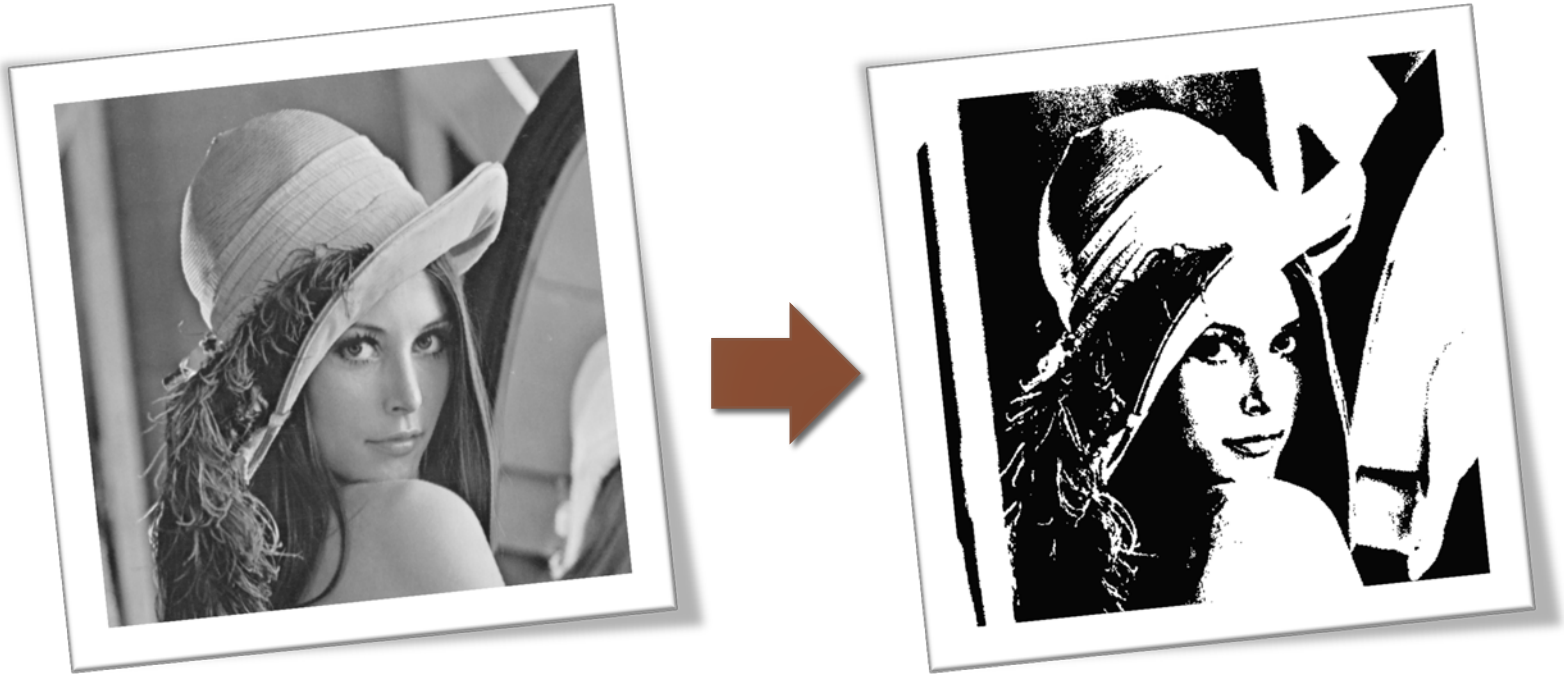
Image Histogram(直方圖)

- grayscale histogram



- 以gray value為x軸的Pixel數量統計圖
  - X軸: gray value
  - Y軸: 像素點數量

Binarize (二値化)

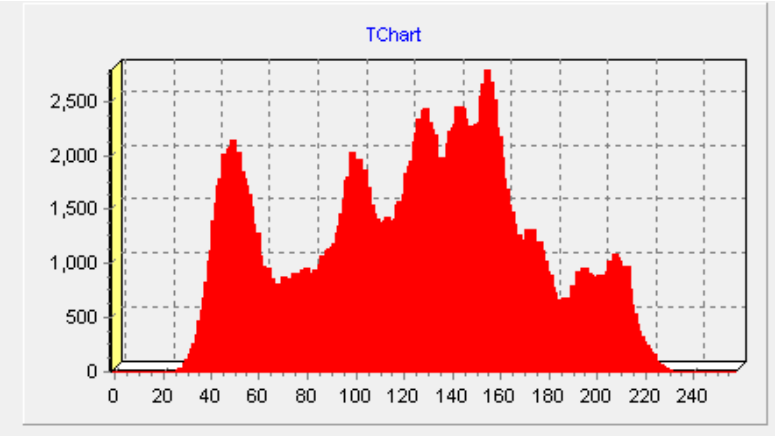


Binarize (二值化)

二值化

- 門檻值 (threshold)

gray value = 255, if gray > threshold
gray value = 0, else.



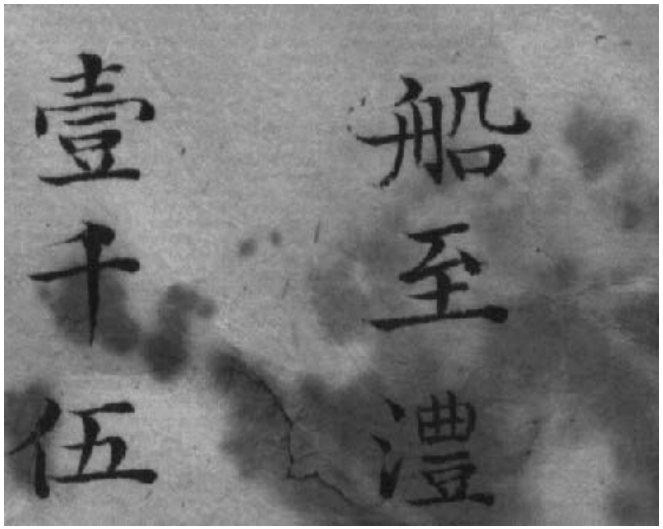
Homework #2.3

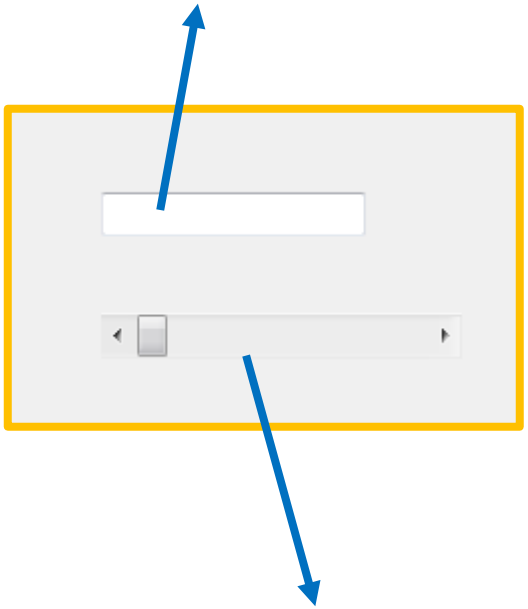
✓實做二值化, 門檻值可由使用者決定

使用HScrollBar 或 TextBox

或 cin (c++)

或 scanf (c)





元件取值

TextBox

```
int tmp1 = Convert::ToInt32(textBox1->Text);     //convert string to 32bit Int
```

```
int value = 10;
textBox1->Text = Convert::ToString(value);     //show value
```

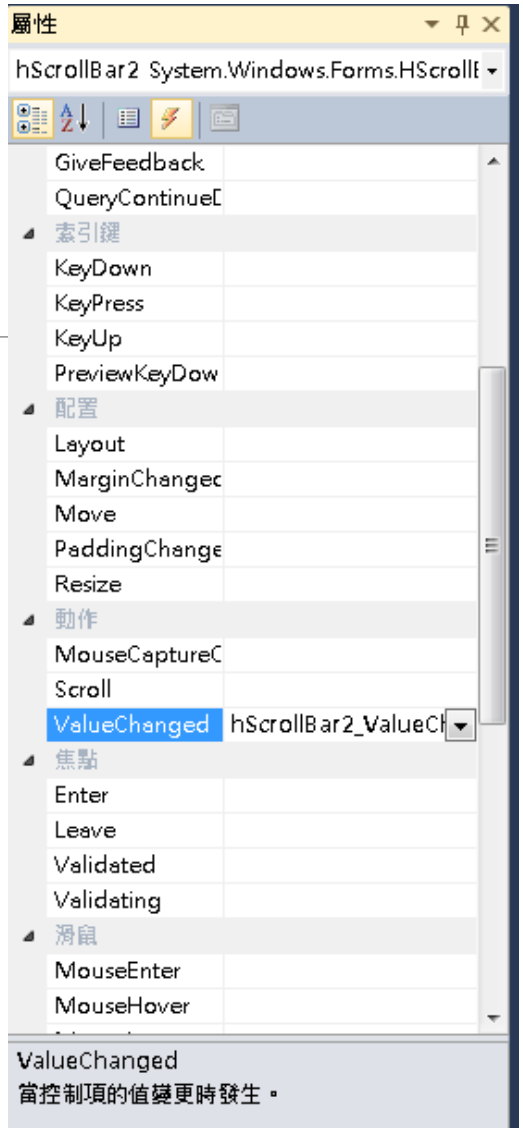
元件取值

HScrollBar

使用ValueChanged事件

拉動時最右邊為Maximum-LargeChange+1

故設Maximum = 255+LargeChange-1



取值:

```
int tmp2 = hScrollBar1->Value;                    //get value of scrollbar
```