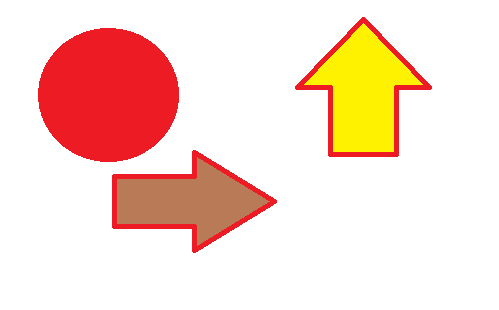
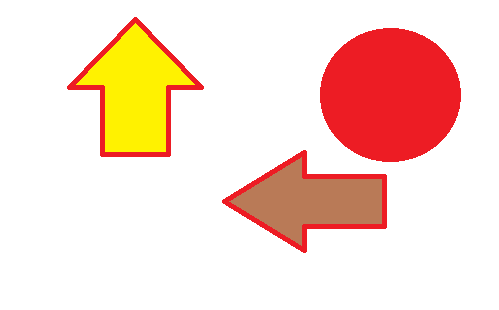
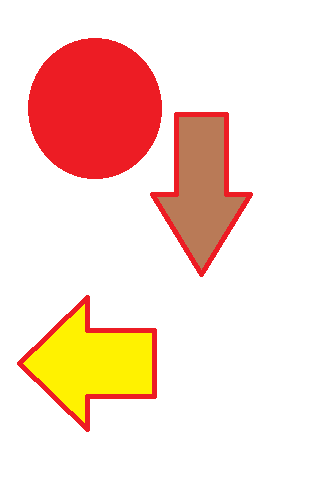
程式設計報告書

1072041 陳泳志

運行結果

原圖

鏡像轉90度

程式解說

typedef struct

{

BYTE b;

BYTE g;

BYTE r;

}RGB;

typedef：定義一個已知資料型態的別名

struct：自訂結構的語法

class CBitmap {

定義一個Cbitmap類別

BITMAPFILEHEADER fileHeader;

BITMAPINFOHEADER infoHeader;

宣告BMP檔案標頭以及資訊標頭變數

ifstream pfin;

ofstream pfout;

ifstream用於讀取，ofstream用於輸出

int offset;

RGB \*\*img;

Offset用來計算BMP的補0，img二維陣列用來儲存像素資料

void load(const char\* path) {

pfin.open(path, ios::binary);

pfin.read((char\*)&fileHeader, sizeof(BITMAPFILEHEADER));

pfin.read((char\*)&infoHeader, sizeof(BITMAPINFOHEADER));

load函式

用二進制打開指定路徑的文件，並讀取檔案標頭以及資訊標頭

offset = (infoHeader.biWidth \* 3) % 4;

if (offset != 0) {

offset = 4 - offset;

}

利用讀到的圖片寬度計算每一行最後應該補幾個0

img = new RGB\*[infoHeader.biHeight];

for (long i = infoHeader.biHeight - 1; i >= 0; i--) {

img[i] = new RGB[infoHeader.biWidth];

for (long j = 0; j < infoHeader.biWidth; j++) {

pfin.read((char\*)(&img[i][j]), sizeof(RGB));

}

動態宣告img陣列，並且在宣告一行之後讀入一行圖片

因為BMP的像素資訊是從下到上、從左到右，所以i是反過來的

if (offset != 0) {

char ignore;

for (int k = 0; k < offset; k++) {

pfin.read(&ignore, sizeof(char));

}

}

}

pfin.close();

讀完一行之後，檢查圖片是否有補0，如果有的話就將多出來的0讀入到宣告的ignore裡面

即可繼續讀下一行

void write(const char\* path) {

pfout.open(path, ios::binary);

pfout.write((char\*)&fileHeader, sizeof(BITMAPFILEHEADER));

pfout.write((char\*)&infoHeader, sizeof(BITMAPINFOHEADER));

write函式

用二進制打開或新建掉指定路徑的文件

並將檔案標頭以及資訊標頭寫入

for (long i = infoHeader.biHeight - 1; i >= 0; i--) {

for (long j = 0; j < infoHeader.biWidth; j++) {

pfout.write((char\*)(&img[i][j]), sizeof(RGB));

}

將儲存的img內的像素資訊寫入，一樣要把i反過來

if (offset != 0) {

char ignore=0;

for (int k = 0; k < offset; k++) {

pfout.write(&ignore, sizeof(char));

}

}

}

pfout.close();

寫完一行之後檢查是否需要補0，要的話就寫入0

string fileHeaderToString() {

string temp = "bfOffBits:" + to\_string(fileHeader.bfOffBits) + " bfSize:" + to\_string(fileHeader.bfSize) + " bfType:" + to\_string(fileHeader.bfType);

return temp;

}

string inforHeaderToString() {

string temp = "biSize:" + to\_string(infoHeader.biSize) + " biWidth:" + to\_string(infoHeader.biWidth) + " biHeight:" + to\_string(infoHeader.biHeight) +

" biBitCount:" + to\_string(infoHeader.biBitCount) + " biCompression:" + to\_string(infoHeader.biCompression) + " biSizeImage:" +

to\_string(infoHeader.biSizeImage) + " biClrUsed:" + to\_string(infoHeader.biClrUsed) + " biClrImportant:" + to\_string(infoHeader.biClrImportant);

return temp;

}

兩個顯示檔案標頭以及資訊標頭內容的函示，不多做贅述

void toMirror() {

RGB temp;

for (long i = 0; i < infoHeader.biHeight; i++) {

for (long j = 0, k = infoHeader.biWidth - 1; j < k; j++, k--) {

temp.r = img[i][j].r;

img[i][j].r = img[i][k].r;

img[i][k].r = temp.r;

temp.g = img[i][j].g;

img[i][j].g = img[i][k].g;

img[i][k].g = temp.g;

temp.b = img[i][j].b;

img[i][j].b = img[i][k].b;

img[i][k].b = temp.b;

}

}

}

toMirror函式

首先宣告一個temp用來暫時存放像素資訊，之後從每一行的兩端開始，將像素資料互換  
換到中間代表全部都換過了就可以繼續下一行

void toRotate90() {

int temp = infoHeader.biHeight;

infoHeader.biHeight = infoHeader.biWidth;

infoHeader.biWidth = temp;

infoHeader.biSizeImage = (infoHeader.biWidth \* 24 + 31) / 32 \* 4 \* infoHeader.biHeight;

fileHeader.bfSize = 54 + infoHeader.biSizeImage;

toRotate90函式

也是一樣先宣告一個tempx來暫存資料，之後把資訊標頭的寬高互換

還要透過公式計算biSizeImage以及bfSize

RGB\*\* img2;

img2 = new RGB \* [infoHeader.biHeight];

for (long i = 0; i < infoHeader.biHeight; i++) {

img2[i] = new RGB[infoHeader.biWidth];

for (long j = 0; j < infoHeader.biWidth; j++)

img2[i][j] = img[j][infoHeader.biHeight - 1 - i];

}

動態宣告一個img陣列，因為前面已經把寬高互換了所以這邊就不需要再宣告的時候換

一樣每宣告一行就讀入一行

offset = (infoHeader.biWidth \* 3) % 4;

if (offset != 0) {

offset = 4 - offset;

}

重新計算要不要補0

for (int i = 0; i < infoHeader.biWidth; i++) {

delete[]img[i];

}

img = img2;

將原本的img內容清掉，指定成剛剛宣告的img2

~CBitmap() {

for (int i = 0; i < infoHeader.biHeight; i++) {

delete[]img[i];

}

delete[]img;

}

解構子會把img清除

