

2025_NYCU_OOPFP_Image_Processing Report

113511007 林俊爾

1. Briefly describe your work in Step 1 ~ 5 and attach result images.

Step 1: 加載與展示圖片

在這部分，程式設計為可以加載單張，或整個資料夾的圖片，種類可以選擇灰階與彩色圖片，但若一次加載一整個資料夾，全部都會統一種類。上方會顯示現在加載成功的資料夾。顯示圖片部分，提供使用者 X_server、ASCII、CMD (架空) 三種顯示的選項。

圖片加載過程：

```
IMAGE PROCESSING
OOP Project | ID:113511007

=====
No images loaded.

Choose loading type:
1. Load single image
2. Load all images from a folder
Enter your choice (1-2, or 0 to go back): 2
Enter folder path: Image-Folder
Select image loading mode:
1. Load as RGB
2. Load as Grayscale
3. Auto Detect (try RGB then Grayscale)
Enter your choice (1-3): 1
Image '3-2.jpg' loaded successfully as RGB!
Image '1-2.jpg' loaded successfully as RGB!
Image '2-2.jpg' loaded successfully as RGB!
Image '1-1.jpg' loaded successfully as RGB!
Image 'truck.png' loaded successfully as RGB!
Image '4-1.jpg' loaded successfully as RGB!
Image '4-2.jpg' loaded successfully as RGB!
Image '3-1.jpg' loaded successfully as RGB!
Image '2-1.jpg' loaded successfully as RGB!
Image 'lena.jpg' loaded successfully as RGB!

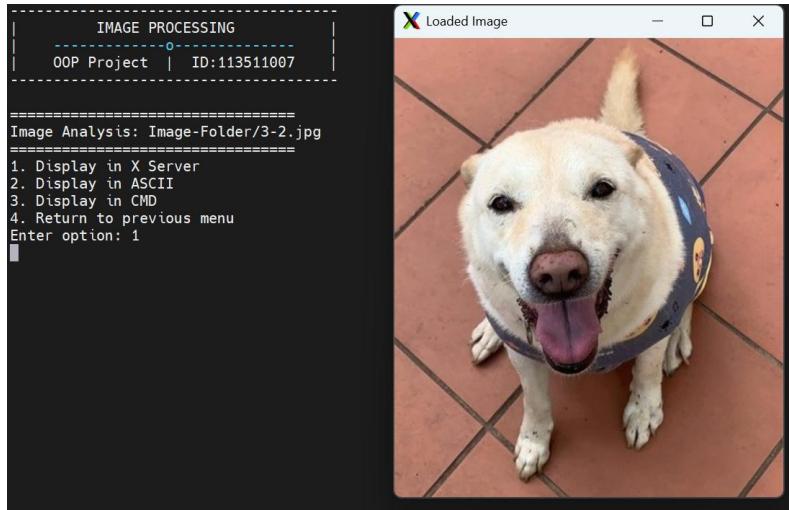
Press Enter to continue...■
```

1. 選擇加載類型（單張/資料夾）

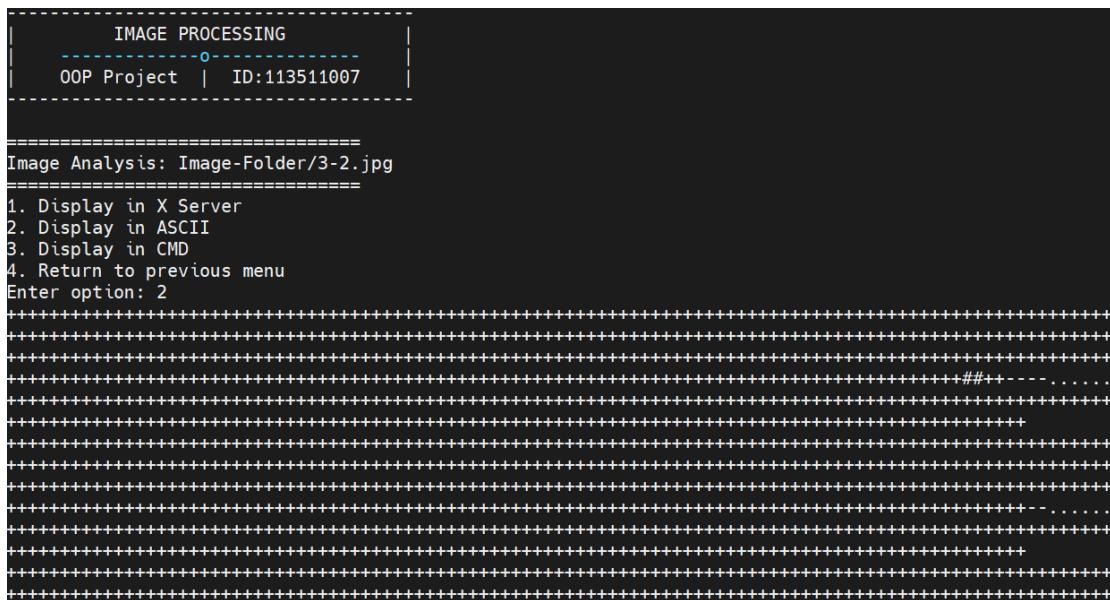
2. 選擇圖片類型（灰階/彩色）

3. 加載成功！

X_server 展示圖片：



ASCII 展示圖片：(尺寸大，難辨認)



Step 2:

此專題中將 image.h、gray_image、rgb_image 三個 class 設定為繼承關係。

關係圖如下：除了專題說明中的指定成員函數、成員變數，也新增了 int

channels;的成員變數以及 int get_channels(); virtual int** gray_get_pixels();

virtual int*** rgb_get_pixels();三個函式以方便後續操作。

Step 3:

這個部分使用者可以挑選自己喜歡的濾鏡，也可以重疊。基礎部分包含

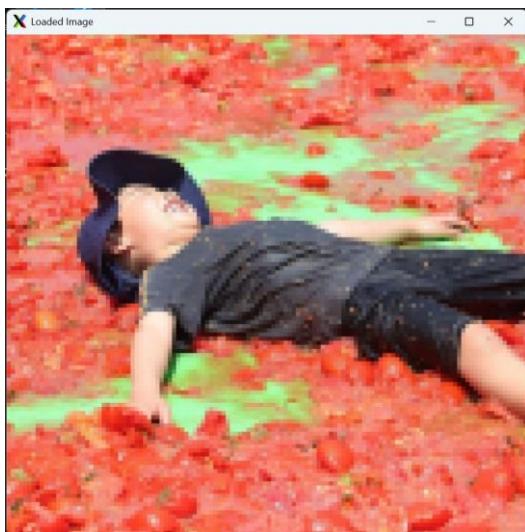
Horizontal Flip、Mosaic filter、Gaussian filter、Laplacian filter、加分部分

包含 fish eye filter、cold/warm adjustment、luminance enhancement。

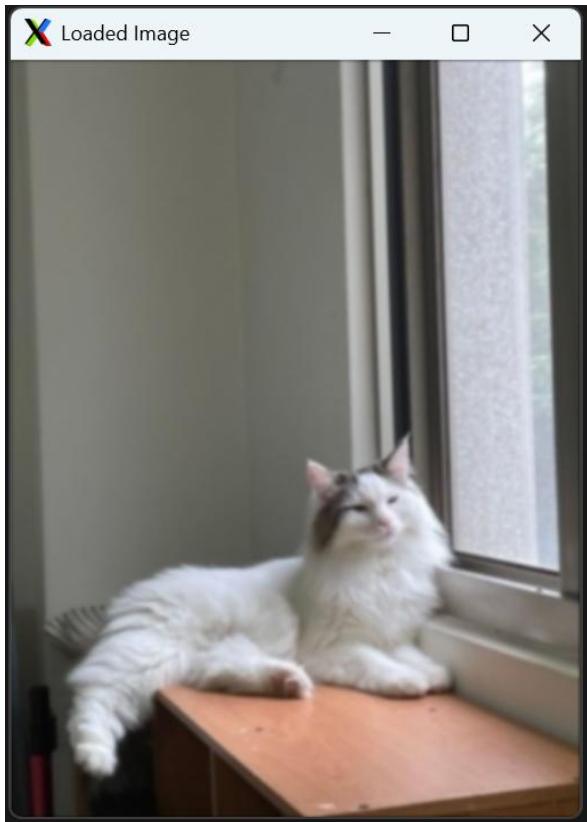
Horizontal Flip:



Mosaic filter:



Gaussian filter:



Laplacian filter:



fish eye filter:



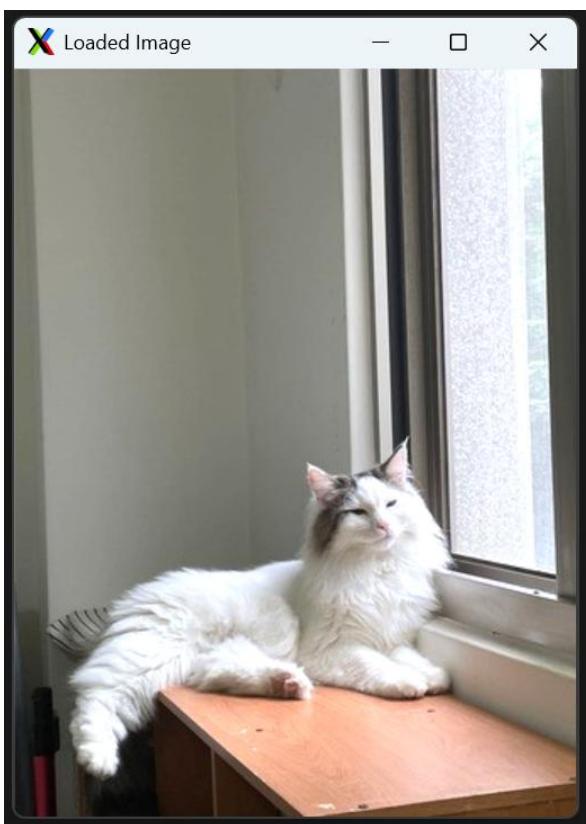
Warm adjustment:



cold adjustment



luminance enhancement



Step 3:

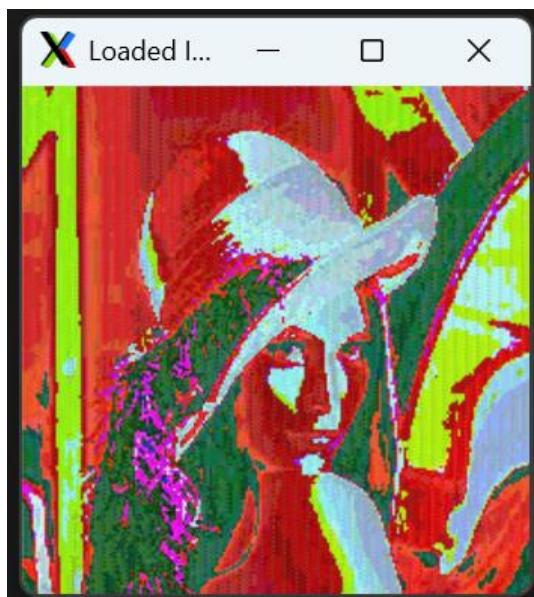
考量到圖片加密的不同特性，這個部分在專題中被分為兩大類。其中一類是將文字隱藏進圖片中，不大幅改動圖片（steganography），如同專題說明的一樣。專題中包含了 LSB 加密法，以及 bit-plane 加密法。Bit-plane 加密法提供更多層空間，讓可隱藏的長度增加，LSB 加密法是 Bit-plane 的特例，將訊息一律藏在第 0 層。而此專題中的 bit-plane 加密法將層數固定在 0、1、2 層，確保能進行盲解。

執行過程：

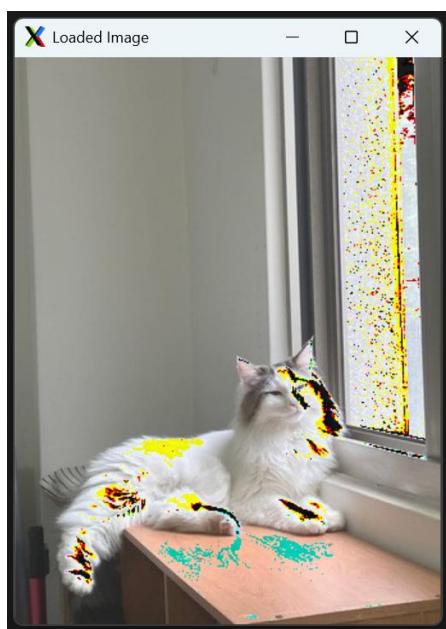
```
-----  
|           IMAGE PROCESSING          |  
|-----o-----|  
| OOP Project | ID:113511007 |  
|-----  
=====  
Steganography Menu  
Image: Image-Folder/2-1.jpg  
=====  
1. LSB Encryption  
2. Bit-Plane Encryption  
3. Return  
Enter option: 1  
Enter message to embed (or 'd' for default - Hello World): d  
Message embedded using LSB!  
Press Enter to continue...  
-----  
|           IMAGE PROCESSING          |  
|-----o-----|  
| OOP Project | ID:113511007 |  
|-----  
=====  
Steganography Menu  
Image: LSB_encrypted_2-1.png  
=====  
1. LSB Decryption  
2. Bit-Plane Decryption  
3. Return  
Enter option: 1  
Decrypting image...  
Embedded message:Hello World  
Press Enter to continue...
```

第二類加密方式會大幅改動圖片，而圖片本身就是要加密過程要隱藏的訊息 (cryptography)。專題中實作了三種加密法，分別是 XOR、Caesar Cipher、Substitution Cipher。其中，Substitution Cipher 提供檔案輸入的方式，讓使用者不用一個一個輸入。

XOR:



Caesar Cipher



Substitution Cipher:



2. Explain where inheritance and polymorphism are used in the entire project.

只有在 image 部分實作，可見上一部份的 step 2

3.Q&A

Please explain roughly what make install does.

make install 會執行安裝流程，通常將第三方函式庫或必要檔案下載、解壓、

編譯，並安裝到指定位置供主程式使用。

How does the makefile help compile this project? (Answer from

inc/src/)?

Makefile 會自動偵測 inc/ 中的標頭檔與 src/ 中的源碼檔案，將其依賴關係

處理後產生對應的 .o 物件檔，最後連結成可執行檔。

Please explain how to design the bit field algorithm in Step 3, how to

decide how many bits to use, and what are the advantages of doing so?

可根據資料最大值決定最少所需 bit 數 (如值小於 16 用 4 bits)。好處是節省記憶體，提高資料儲存與傳輸效率。

In Step 4, what will happen if the length of target string is too long?

如果字串超出緩衝區長度，可能導致記憶體溢位 (buffer overflow)，引發錯誤、當機或資安漏洞。

What are the pros/cons of the encryption method in step 4?

優點是實作簡單、執行快速；缺點是安全性低，容易被暴力破解或頻率分析破解。

Use valgrind to perform dynamic analysis on your program, and show the execution results and explain the differences between 5 types of memory leak.

```
|-----|  
| IMAGE PROCESSING |  
|-----|  
| OOP Project | ID:113511007 |  
|-----|  
=====  
1. Load Images  
2. Enter Command Mode  
3. Exit System  
Enter option: 3  
Exiting system...  
==11998==  
==11998== HEAP SUMMARY:  
==11998==     in use at exit: 0 bytes in 0 blocks  
==11998==   total heap usage: 4,760,490 allocs, 4,760,490 frees, 100,521,925 bytes allocated  
==11998==  
==11998== All heap blocks were freed -- no leaks are possible  
==11998==  
==11998== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)  
202400P015/fp> █
```

Definitely lost：程式遺失指標，無法釋放記憶體。

Indirectly lost：主指標遺失導致附屬記憶體無法釋放。

Possibly lost：指標可能指向多個位置，Valgrind 不確定。

Still reachable：記憶體尚有指標指向，但未釋放。

Suppressed：被過濾掉的已知問題，不顯示。

4. Bugs and resolution:

此部分挑選製作專題中四個較重大的 bug。由於當下並未截圖，以下用文字敘述。

Bug 1: 忽略 jpg 的壓縮特性。如果將加密後的圖片在結束程式前直接解密可正常運行，因為這時還是完整的像素矩陣。但只要離開程式，系統將加密過的像素矩陣存成 jpg 檔下載，那麼就沒有辦法解密了。

Resolution: 將涉及加密的圖片一律存成 png 檔，確保資訊有被完整保存。

Bug 2：圖片加載路徑

原先想讓使用者輸入路徑，讓程式自動判斷要到加載一整個資料夾或是單一圖片。但是容易判斷錯誤，導致程式中止，但事實上不一定是路徑的問題。使用者難以得知錯誤是發生在路徑錯誤、檔案不存在，還是圖片格式錯誤。

Resolution:

重寫 load_image_menu()，改為先詢問載入類型（單圖 / 資料夾），再詢問圖片格式（RGB / Gray / Auto），並分開處理路徑錯誤與圖片讀取錯誤，給出具體錯誤訊息，有助除錯與使用者操作。

Bug 3: 圖片下載。這部分遇到的主要問題是檔名與路徑處理不當。最初直接使用原始路徑儲存，導致系統因非法字元（如 /, :, *）而無法寫入檔案，且不同處理階段的檔名容易混淆。例如：filtered_Image-Folder/1-1.jpg 會導致錯誤，

```
[CImg] *** CImgIOException *** cimg::fopen(): Failed to open file  
' filtered _Image-Folder/1-1.jpg' with mode 'wb'.  
  
terminate called after throwing an instance of  
'cimg_library::CImgIOException'  
  
what(): cimg::fopen(): Failed to open file 'photomosaic_Image-  
Folder/1-1.jpg' with mode 'wb'.  
  
Abort (core dumped)
```

無法正確把處理過的圖片下載到 processed_Image-Folder 中。

Resolution: 從完整路徑中提取純檔名，去除代表資料夾的部分，例如將 filtered_Image-Folder/1-1.jpg 轉換為 filtered_1-1.jpg

Bug 4: 記憶體釋放錯誤。

這個部分的錯誤相比平常上機的練習還難察覺，因為檔案與函式都相當多，指標的追蹤需要非常仔細。例如：在對圖片進行各式各樣的處理時，為了保留原圖，特意另宣告一個指標存取，但這個指標的釋放就疏忽了。此外，valgrind

無法識別的函式相當多，甚至有時候只是多套幾層濾鏡程式就崩潰了。

Resolution: 記憶體的釋放追蹤只能照著 valgrind 的指示找錯誤的地方，這也是花最久時間的。至於 valgrind 無法識別的函式，起初只有將 X_server 等較好確認的函式註解掉，但問題還是無法解決，於是直接修改 makefile
“OPTFLAGS = -march=native -flto -funroll-loops -finline-functions -ffast-math -O3 ” 為 “ OPTFLAGS = -march=core2 -O2 -g” 。

5.Special features of your program (bonus!):

- 一、增加四種濾鏡：fisheye filter、cold/warm adjustment、luminance enhancement (見 step 3)
- 二、增加四種加密法，包含一種嵌入文字加密：bit-plane 加密法，以及三種圖片加密：XOR、Caesar Cipher、Substitution Cipher (見 step 4)

三、使用者友善介面：

1. 使用簡單易懂的選單，讓使用者能輕易上手。

The screenshot shows a terminal window titled "IMAGE PROCESSING". At the top, it displays "OOP Project | ID:113511007". Below this, a list of current loaded images is shown, each with its name, dimensions, and color space (RGB). A red box highlights this list. To the right, a callout box says "顯示可以處理的圖片" (Display images that can be processed). The menu then lists various processing options: 1. Image Analysis Tools, 2. Filter Processing, 3. Image Encryption (Steganography), 4. Image Decryption (Steganography), 5. Image Encryption (Cryptography), 6. Image Decryption (Cryptography), 7. Save and Return, 8. Remove an Image, and 9. Clear All Images. The option "Enter option: 2" is highlighted with a red box. Below it, "Select image (1-10): 8" is also highlighted with a red box. To the right, another callout box says "選擇想要處理的方法與目標圖片，一次處理一張" (Select the method and target image to process, process one at a time).

```
IMAGE PROCESSING
OOP Project | ID:113511007

=====
Current loaded images:
1. 3-2.jpg (360x425 RGB)
2. 1-2.jpg (292x180 RGB)
3. 2-2.jpg (630x630 RGB)
4. 1-1.jpg (333x443 RGB)
5. truck.png (32x32 RGB)
6. 4-1.jpg (333x443 RGB)
7. 4-2.jpg (360x425 RGB)
8. 3-1.jpg (563x374 RGB)
9. 2-1.jpg (225x225 RGB)
10. lena.jpg (225x225 RGB)
=====

1. Image Analysis Tools
2. Filter Processing
3. Image Encryption (Steganography)
4. Image Decryption (Steganography)
5. Image Encryption (Cryptography)
6. Image Decryption (Cryptography)
7. Save and Return
8. Remove an Image
9. Clear All Images
Enter option: 2
Select image (1-10): 8
```

2. 大量使用 press_enter_to_continue 以及 clear 來整理版面，避免雜亂。
3. 圖片處理後自行加載，使用者可以直接下載到 Processed_Image-Folder。

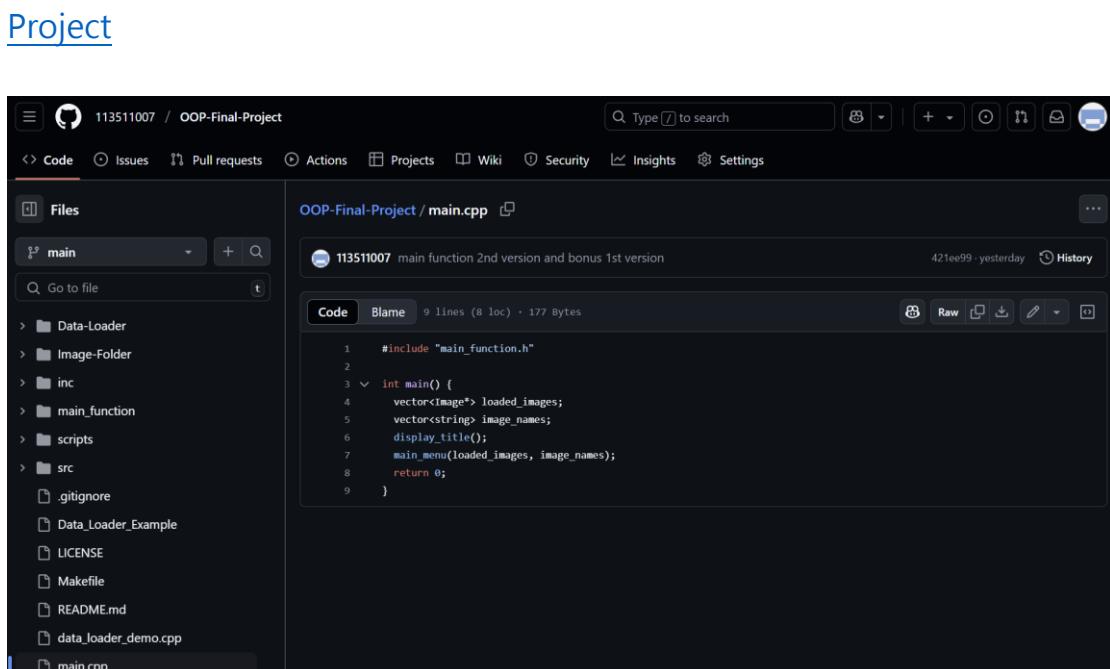
The screenshot shows a terminal window displaying a list of processed images. The list includes the original images from the previous screenshot, plus two new ones: "filtered_3-1.jpg" and "Caesar_Cipher_encrypted_2-2.png". A red box highlights the entire list of images.

```
6. 4-1.jpg (333x443 RGB)
7. 4-2.jpg (360x425 RGB)
8. 3-1.jpg (563x374 RGB)
9. 2-1.jpg (225x225 RGB)
10. lena.jpg (225x225 RGB)
11. filtered_3-1.jpg (563x374 RGB)
12. Caesar_Cipher_encrypted_2-2.png (630x630 RGB)
```

4. 部分函式提供使用者調整參數的管道，讓圖像處理更客製化。

```
-----  
| IMAGE PROCESSING |  
| OOP Project | ID:113511007 |  
-----  
=====Filter Processing: 3-2.jpg  
Current active filters: None  
=====  
1. Horizontal Flip  
2. Mosaic (Block size: 5)  
3. Gaussian Blur (SD: 1)  
4. Laplacian Sharpen  
5. Fisheye Effect  
6. Cold Adjustment (Intensity: 30, Luminance: Keep)  
7. Warm Adjustment (Intensity: 30, Luminance: Keep)  
8. Luminance Enhancement (20%)  
9. Set Filter Parameters  
10. Preview Effects  
11. Apply Filters  
12. Clear all filters  
13. Return to Previous Menu  
Enter an option (1-13):
```

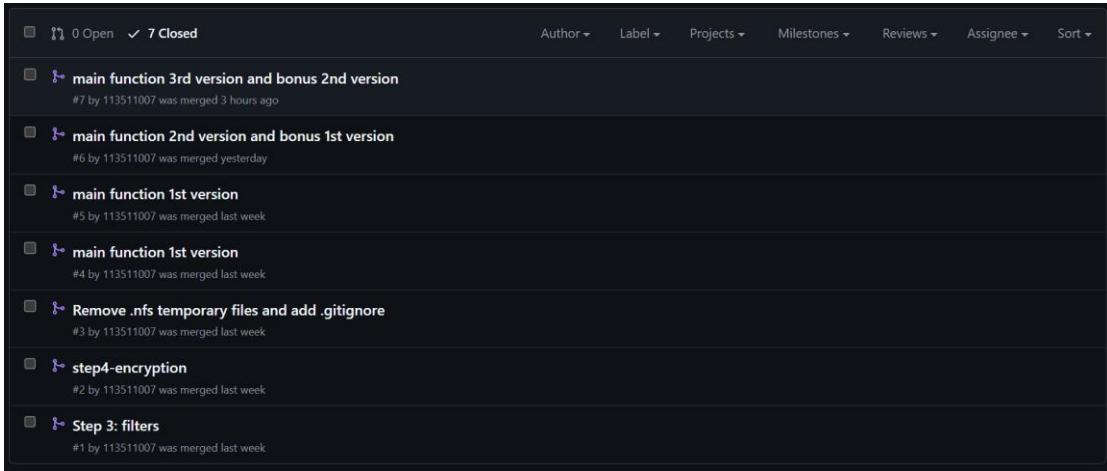
四、使用 github 管理專案：<https://github.com/113511007/OOP-Final-Project>



The screenshot shows a GitHub repository interface for the project "OOP-Final-Project". The repository owner is "113511007". The "Code" tab is selected, displaying the contents of the "main.cpp" file. The code is as follows:

```
#include "main_function.h"  
  
int main() {  
    vector<Image*> loaded_images;  
    vector<string> image_names;  
    display_title();  
    main_menu(loaded_images, image_names);  
    return 0;  
}
```

實作合併分支：



6.Briefly explain how you divided the work and what percentage each person contributed.

獨自完成

7.Feedback & Conclusion.

在這次的專題中，我成功實作了圖片的濾鏡以及加密解密處理，這是我生涯第一次做這麼大型的程式作業。過程中，許多地方非常複雜難懂，就連作為收尾的記憶體釋放檢查都花了我許多時間，原本想要做更多加分項，但時間卻連為基本部分的程式碼除錯都快不夠，因此，我想我還有很多需要精進的地方。雖然這個專題不像其他題目是偏向動態、遊戲類的，不過，實際做出來後還是非常有成就感，也收穫很多。很感謝助教用心地出題，讓我有了這樣的學習經驗。