VNU HCMC-UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

REPORT
THE BITMAP PROCESSING PROJECT

COURSE: PROGRAMMING TECHNIQUES

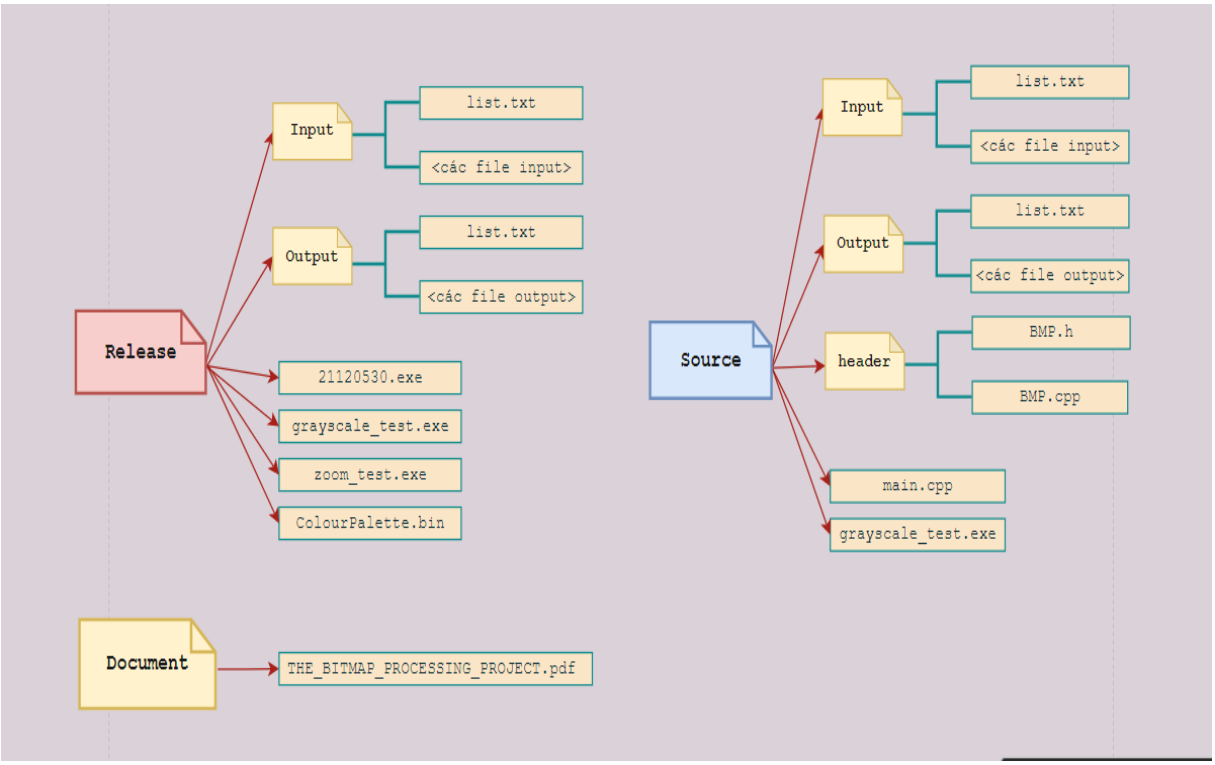
21120530 - Nguyễn Hoàng Phúc

Class: 21CTT5

Mục lục

1 Giới thiệu

1.1 Cấu trúc folder



1.2 Ý nghĩa các file và các folder

Input

Chứa list.txt(bao gồm toàn bộ tên file test có sẵn trong folder input) và danh sách các file input.

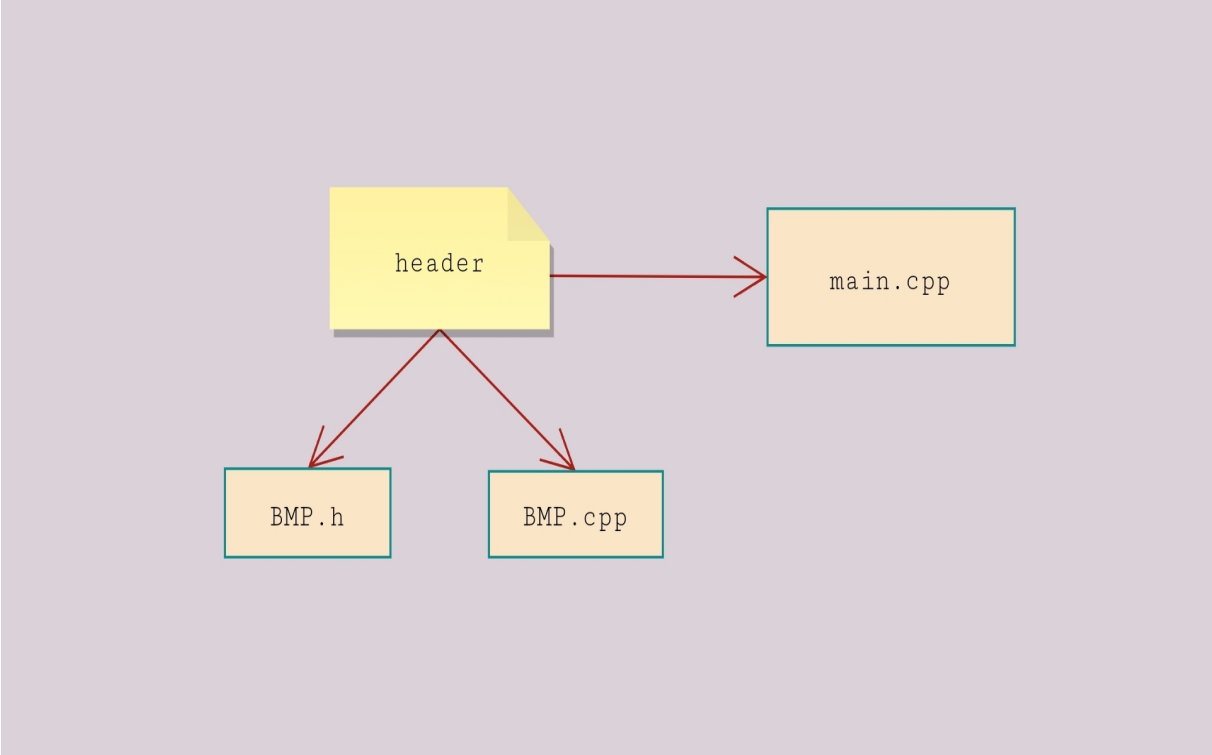
Output

Chứa list.txt(bao gồm toàn bộ tên file output có trong folder output(test)) và là nơi sẽ chứa các file test có sẵn sau khi chạy chương trình viết sẵn (Grayscale.exe và Zoom.exe).

Release

Chứa source code của project, bao gồm main.cpp, BMH.h, BMP.cpp, các file test input, output, bảng màu ColourPalette.bin, file test dựng sẵn như zoom_test.exe, grayscale_test.exe.

1.3 Liên kết file



1.4 Cách compile thủ công

Để build project trong vscode terminal, chạy lệnh:

```
g++ main.cpp Header\BMP.cpp -o main.exe
```

1.5 Hướng dẫn chạy file bằng tham số dòng lệnh

Giả sử ta đang ở cấp Source

Để chuyển ảnh 32/24bit sang 8bit, chạy lệnh:

```
21120530.exe -conv <File_Input> <File_Output>
```

Để thu nhỏ ảnh 8/24/32bit theo tỉ lệ S, chạy lệnh:

```
21120530.exe -zoom <File_Input> <File_Output> <S>
```

2 Ý nghĩa từng hàm

void AutoTest(int s)

Hàm chạy thử hàm zoom (theo tỉ lệ s) với Input là các file test có sẵn trong

folder Input. File ảnh sau khi thực hiện xong sẽ được đưa vào folder Output.
Có thể chạy file bằng command prompt với lệnh sau:

```
zoom.exe <S>
```

void AutoTest()

Hàm chạy thử hàm convert với Input là các file test có sẵn trong folder Input. File ảnh sau khi thực hiện xong sẽ được đưa vào folder Output.
Có thể chạy file bằng cách click đúp chuột vào file .exe hoặc sử dụng command prompt với lệnh sau:

```
Grayscale.exe
```

void read(Image &img, const char *fileName)

Hàm đọc dữ liệu của một bức ản bitmap có tên là <fileName> và lưu dữ liệu lưu vào biến img.

void write(Image img, const char *file_name)

Hàm ghi dữ liệu của ảnh bitmap được lưu trong biến img vào file có tên <fileName>.

Pixel **getPixel(Image img)

Hàm chuyển dữ liệu điểm ảnh được lưu trong biến img thành mảng pixel 2 chiều. Kết quả trả về là địa chỉ đầu tiên của mảng. Hàm sẽ thực hiện tuần tự như sau:

- Tính toán padding byte.
- Cấp phát vùng nhớ cho mảng 2 chiều width*height*sizeof(Pixel) byte.
- Đọc dữ liệu điểm ảnh và lưu vào mảng 2 chiều pixel.
- Trả về dữ liệu của mảng 2 chiều.

Image convert(const Image src)

- Hàm chuyển dữ liệu ảnh được lưu trong biến src thành ảnh 8bit vào trong biến newImg. Hàm trả về dữ liệu ảnh 8bit.

Cách cài đặt code.
- Đầu tiên, ta khởi tạo thông tin ảnh đích (newImg). Sau đó chép phần Header, DIB và phần unused(nếu có).
- Chỉnh sửa lại thông tin của compression = 0 (biến đặc trưng cho độ nén dữ liệu, 0 là không nén) và bit per pixel = 8.
- Đọc thông tin bảng màu và lưu vào biến ColorPalette. Vì cấu trúc của file 8bit là Header-DIB-ColorPalette-ImageData, mà trong file 32/24bit không có ColorPalette nên ta phải đọc từ file ColourPalette.bin để lưu vào.

- Cấp phát vùng nhớ cho biến data để lưu dữ liệu điểm ảnh sau khi chuyển.
- Khởi tạo và cấp phát các biến mảng Pixel 2 chiều để thuận tiện cho việc xử lý.
- Duyệt trên data gốc, tính trung bình màu tại pixel[x][y] (R + G + B)/3 và lưu vào pixel[x][y].rgb
- Chuyển dữ liệu từ mảng 2 chiều (pixel[x][y]) thành mảng char* một chiều.
- Ghi ảnh ra file.

Pixel avg(Pixel *pixel, int16_t size)

Hàm tính trung bình giá trị từng màu (R G B) của mảng một chiều pixel với size điểm ảnh rồi trả về một điểm ảnh mới.

Image zoom(const Image src, int s)

Hàm thu nhỏ ảnh theo tỉ lệ s cho trước, áp dụng cho ảnh 24bit hoặc 32bit.

- Cách cài đặt code. (để code đơn giản, dễ nhìn nên tài liệu sẽ lược bớt thuộc tính header, DIB của struct Image)

Khởi tạo thông tin ảnh đích (newImg). Sau đó tính toán lại kích thước ảnh với:

- newImage.width = srcImage.width / s
- newImage.height = srcImage.height / s
- padding = (4 - (newImg.width * (newImg.bit_per_pixel / 8)) % 4) % 4;
- file_size
- image_size

Vì một số ảnh có kích thước không chia hết cho s dẫn đến việc tính toán trên pixel gặp khó khăn nên ta sẽ tăng kích thước ảnh gốc lên sao cho vừa đủ chia hết cho s.

```
while (newImg.dib.width % s != 0)
    newImg.dib.width++;
while (newImg.dib.height % s != 0)
    newImg.dib.height++;
```

Các pixel thừa ra sẽ được bỏ qua khi duyệt và tính toán nên sẽ không bị lỗi vùng nhớ không xác định bằng câu lệnh if.

```
if ((x + i) < src.dib.height
 && (y + j) < src.dib.width)
{
```

```
            temp[count] = srcPix[x + i][y + j];
            count++;
        }
```

Cấp phát vùng nhớ cho biến data để lưu dữ liệu điểm ảnh sau khi chuyển.

Khởi tạo và cấp phát các biến mảng Pixel 2 chiều để thuận tiện cho việc xử lý.

Duyệt trên data gốc, tính trung bình từng màu theo ma trận con SxS thuộc ma trận pixel gốc.

Ghi ảnh ra file.

Image zoom(const Image src, int s, int bpp)

Hàm có chức năng tương tự như hàm Image zoom(const Image, int) nhưng áp dụng cho file 8bit.

Cách xây dựng code cũng tương tự như hàm trên. Chỉ khác ở chỗ ta thêm ColourPalette theo thứ tự cấu trúc ảnh 8bit khi ghi file: Header-DIB-ColourPalette-ImageData.

void delPixel(Pixel **&pixel, int height)

Hàm hủy bộ nhớ của mảng 2 chiều Pixel chứa dữ liệu điểm ảnh.

void delImg(Image &img)

Hàm hủy các mảng động có trong struct Image như data, unused, color-Palette.

3 Reference

Cấu trúc file 8bit

Đọc và ghi file