CSED101. Programming & Problem solving
Fall 2022

Programming Assignment #2
(75 points)

문새미 (saemi@postech.ac.kr)

■ **제출 마감일: 2022.11.15 23:59**

■ **개발 환경: Windows Visual Studio 2019**

■ **제출물**

- C 소스 코드 (assn2.c)
  - ➢ 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn2.docx, assn2.hwp 또는 assn2.pdf)
  - ➢ 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
  - ➢ 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
  - ➢ 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ **주의 사항**

- 컴파일이나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 "POSTECH 전자컴퓨터공학부 부정행위 정의"를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조하세요).
- 이번 과제는 추가 기능 구현과 관련된 추가 점수가 따로 없습니다.

[들어가기 전]

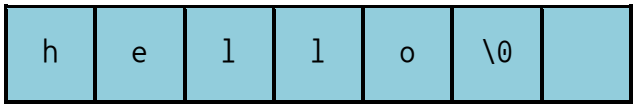
1. 문자열(string)

- 연속된 문자들로 C 언어에서 문자열 앞 뒤에 " "를 이용한다.
- char 형의 1차원 배열을 이용하여 문자열을 저장한다.
- 배열에 문자열을 저장할 때는 끝을 NULL 문자 ('\0')를 넣어서 표시한다.

2. 선언

- **char str[] = "hello";**
  위와 같이 선언과 동시에 초기화를 하게 되면, 자동으로 문자열의 끝에 널문자가 추가된다.

  str



3. 입출력 예시

아래는 문자열 입출력 예시 및 파일명을 입력 받아 해당 파일을 여는 예시입니다.

```
char filename[75]; // 최대 74자 까지 저장 가능
FILE* fp;

printf("Enter input filename: ");
scanf("%s", filename); // 공백 전 까지만 읽음
printf("%s", filename); // 입력 받은 문자열을 화면에 출력

fp = fopen(filename, "r"); // 입력 받은 파일을 읽기 모드로 열기
```

<실행 예시> (아래의 빨간색 밑줄은 사용자 입력에 해당)

```
Enter input filename: mushroom.ppm
mushroom.ppm
```

■ Problem: 이미지 색조 변경 프로그램

(목적)

- 다차원 배열의 선언과 사용을 익힙니다.
- 텍스트 파일을 통한 입출력을 익힙니다.

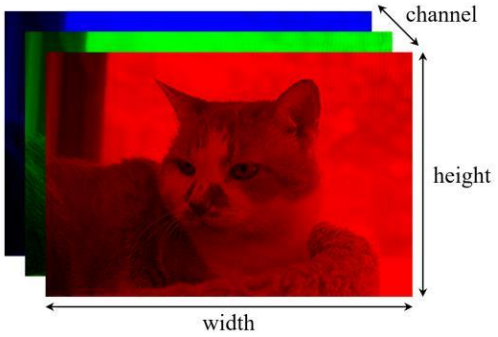
(주의사항)

1. 이번 과제는 3차원 배열을 선언하고 사용하는 과제이므로, 이미지는 반드시 3차원 배열로 선언 후 사용해야 합니다.
2. 문서에 반드시 정의해서 사용해야 할 사용자 정의 함수가 설명되어 있으니 확인 후 구현하도록 합니다. 이 때, 설명에서 지정한 사용자 정의 함수의 매개변수의 개수와 자료형, 함수 이름, 반환 자료형 등은 자유롭게 변경이 가능합니다. 그러나 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 무엇을 어떻게 변경해서 구현했는지 보고서에 기록하도록 합니다. 이외에 필요한 함수는 정의해서 사용할 수 있습니다.
3. 프로그램 구현 시, `main()` 함수를 호출을 직접 하지 않습니다. 즉, 소스 코드 내에 `main();` 이라고 호출하지 않습니다.
4. 전역 변수, `goto` 문, `string` 관련 함수, 구조체는 사용할 수 없으며, 포인터의 경우 수업 시간에 다룬 내용에 한해서 사용이 가능합니다.
5. 사용자 입력에서 숫자를 입력 받는 부분에는 숫자만 입력하는 것으로 가정합니다. 즉, 숫자 입력 받는 부분에는 문자 등의 입력에 대해서는 고려할 필요가 없습니다.
6. 명시된 에러 처리 외에는 고려하지 않아도 됩니다.
7. 문제의 출력 형식은 채점을 위해 아래의 실행 예시와 최대한 비슷하게 작성해 주세요.

I. 용어 및 개념 설명

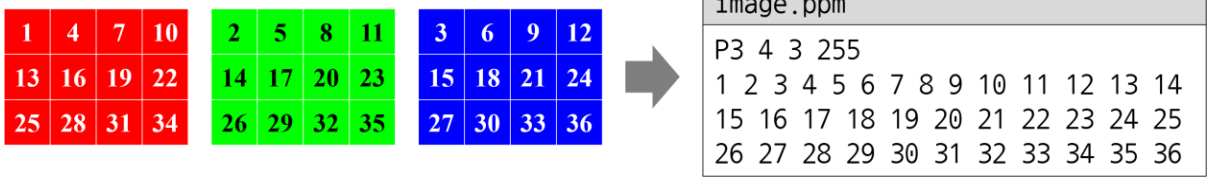
(이미지)

컬러 이미지는 너비(width), 높이(height), 색상 채널(channel) 3 차원으로 구성된 배열로 표현됩니다. 일반적으로 이미지는 빛의 3 원색인 빨강, 초록, 파랑으로 구성되어 총 3 개의 채널을 가지고 있습니다. 많은 이미지들이 효율성을 위해 각 데이터를 1 Byte 로 저장하기 때문에, 한 픽셀 의 값은 0 부터 255 사이의 정수 3 개로 표현됩니다. 이러한 규칙에 따르면 (0, 0, 0)은 검정색, (255, 255, 255)는 흰색, (255, 0, 0)은 빨간색 등으로 표현할 수 있습니다.



(PPM 이미지 파일 포맷)

실생활에서 흔히 접할 수 있는 PNG, JPEG 이미지의 경우 압축된 바이너리 파일로 용량이 작고 효율성이 높지만, 구조의 복잡성 때문에 과제에 사용하기엔 부적합합니다. 따라서 과제에서는 텍스트 형식으로 구성된 PPM(Portable PixMap format) 이미지 포맷을 사용합니다. PPM 이미지의 예시를 보면서 포맷 구조를 알아보겠습니다.



PPM 이미지에서 가장 먼저 나오는 ‘P3’은 하나의 픽셀이 3 개의 데이터로 이루어져 있다는 뜻이며, 그 뒤로 나오는 ‘4 3’은 각각 이미지의 너비와 높이를 뜻합니다. 그 뒤의 ‘255’는 각 데이터 값이 0~255 사이의 정수로 표현됨을 뜻합니다. 과제에 사용할 컬러 이미지는 항상 첫 번째 값이 ‘P3’이며 네 번째 값은 ‘255’ 입니다.

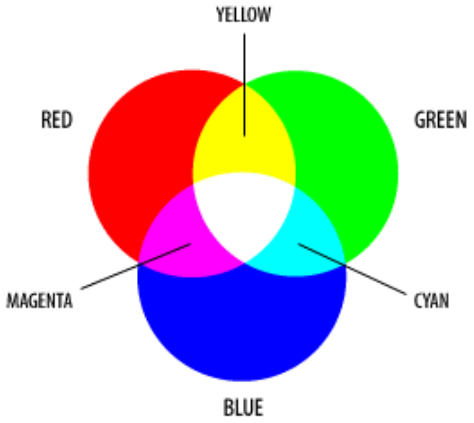
나머지는 각각의 픽셀 데이터가 채널, 너비, 높이 순으로 나열되어 있습니다. 예를 들어 위의 예제에서 첫 4 개의 값을 제외한 가장 앞의 ‘1 2 3’은 왼쪽 위 꼭짓점에 위치한 픽셀의 RGB 값 을 뜻하고, 이어지는 ‘4 5 6’은 그 오른쪽 픽셀의 RGB 값을 뜻합니다. 제공되는 ppm 파일은 ASCII code 로 작성되었기 때문에 텍스트 편집기로 픽셀 값을 확인할 수 있습니다.

(RGB, HSV 색공간(Color space))

색공간은 주로 세자리 혹은 네자리의 숫자 또는 문자를 통해 색을 일정한 규칙으로 정리해 놓은 공간입니다. 색공간은 어떤 색 모델을 활용하는지에 따라 종류가 다양합니다. 이번 과제에서는 RGB 와 HSV 라는 색공간을 사용할 예정입니다.

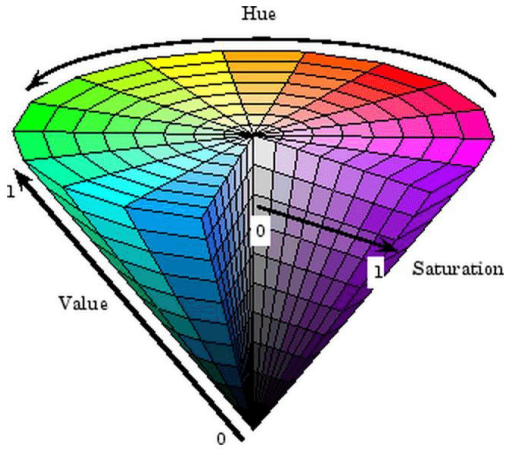
- **RGB 색공간**

  RGB 는 각각 Red, Green, Blue 를 의미하며, 이 3 가지 색을 가산혼합 하여 색을 표현하는 방식이다. 디지털 공간에서의 색은 대부분 RGB 색공간에서 정의되며, 각 R, G, B 값은 0 에서 255 사이의 정수 값을 가집니다.



- **HSV 색공간**

  HSV 는 각각 Hue(색상), Saturation(채도), Value(명도)의 약자로 아래의 그림과 같이 원뿔 모양으로 색공간을 표현한다. H 는 0 에서 360 사이의 값, S 와 V 는 0 에서 1 사이의 값을 가집니다. 특히 S 는 색의 진함 정도를 나타내는데, 0 인 경우에는 흰색과 검정색과 같은 무채색을 나타냅니다.



이번 과제에서 이미지를 출력할 때는 RGB 색공간을, 색조를 변경할 때는 HSV 색공간을 사용하기 때문에, 두 색공간 사이의 값을 변환하기 위한 공식이 필요합니다. 아래의 공식을 활용하여 과제에 사용하시면 됩니다.

- **RGB to HSV**

각 *R, G, B* 값은 0 에서 255 사이의 정수 값을 가진다.

$$R' = R/255$$
$$G' = G/255$$
$$B' = B/255$$

위 공식 적용한 각 *R', G', B'* 값은 0 에서 1 사이의 실수 값을 가진다.

$$C\max = \max(R', G', B')$$
$$C\min = \min(R', G', B')$$
$$\Delta = C\max - C\min$$

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta}\right) & , C_{\max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{\max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{\max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{\max} = 0 \\ \frac{\Delta}{C_{\max}} & , C_{\max} \neq 0 \end{cases}$$

$$V = C\max$$

$$\text{If H} < 0 \text{ then, H} = \text{H} + 360^\circ$$

H 를 코드로 작성시, 괄호 부분의 연산이 먼저 수행되도록 반드시 괄호를 넣어주세요 ☺

- **HSV to RGB**

아래의 식을 구현할 시 절댓값과 mod 함수는 각각 “ math.h ”의 **fabs(a, b)** 함수와 **fmod(a, b)** 함수를 사용하시면 됩니다. 변환된 H, S, V 값은 아래의 범위를 가진 실수 값 입니다.

$$\text{Input: } 0 \le H < 360, 0 \le S \le 1, 0 \le V \le 1$$

$$C = V \times S$$
$$X = C \times (1 - |(H/60^\circ) \bmod 2 - 1|)$$
$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \le H < 60^\circ \\ (X, C, 0) & , 60^\circ \le H < 120^\circ \\ (0, C, X) & , 120^\circ \le H < 180^\circ \\ (0, X, C) & , 180^\circ \le H < 240^\circ \\ (X, 0, C) & , 240^\circ \le H < 300^\circ \\ (C, 0, X) & , 300^\circ \le H < 360^\circ \end{cases}$$

$$(R, G, B) = \big((R' + m) \times 255, (G' + m) \times 255, (B' + m) \times 255\big)$$

(콘솔 화면에 픽셀 색 출력 및 환경 설정)

콘솔에 색을 출력하기 위해서는 Escape 코드("\033")를 활용해야 합니다. 이를 활용한 함수는 아래와 같으며, set_color_rgb(r, g, b)에 각각에 해당하는 r, g, b 값을 입력하여 색깔을 설정하면, 그 뒤의 모든 글자는 지정된 색으로 콘솔에 출력이 됩니다. 또한, reset_color()를 호출하면 글자에 저장된 색이 원래대로 돌아옵니다.
아래의 코드는 r, g, b 값에 따라 어떤 색의 글자가 출력 되는지 보여주는 예시 코드이며, <Figure 1>은 예시 코드의 결과를 보여줍니다. 또한, 콘솔에 각 픽셀을 출력하는 기호는 '■'로 합니다.
아래 코드는 맥에서 사용이 가능합니다. **윈도우에서 안 되는 경우, 노란색 형광으로 표시한 부분의 주석을 모두 풀어서 사용합니다.**

```c
#include <stdio.h>
//#include <windows.h> // GetConsoleMode () 등의 사용을 위해 포함

/*int set_vt_mode() {
    DWORD l_mode;
    GetConsoleMode(GetStdHandle(STD_OUTPUT_HANDLE), &l_mode);
    SetConsoleMode(GetStdHandle(STD_OUTPUT_HANDLE), l_mode | 0x0004 | 0x0008);
}*/

void set_color_rgb(int r, int g, int b) {
    printf("\033[38;2;%d;%d;%dm", r, g, b);
}

void reset_color() {
    printf("\033[0m");
}

int main()
{
    //set_vt_mode();
    for (int i = 1; i < 20; i++)
    {
        set_color_rgb(255 - i*10, 255 - i*12, i*9); // 색깔 설정, r g b
        printf("■■■■■■■■■■■■■[r:%3d, g:%3d, b:%3d]\n" , 255-i*10, 255-i*12, i*9);
        reset_color();
    }
    return 0;
}
```

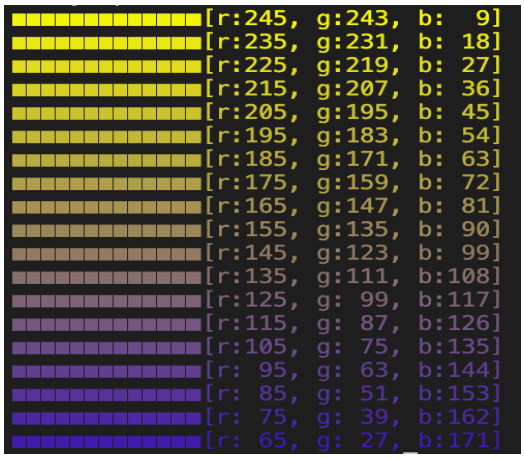


Figure 1 예시 코드 실행 결과

II. 구현 요구사항

이번 과제에서 구현해야 할 것은 다음과 같습니다.

- PPM 이미지를 불러오거나 저장하며 출력합니다.
- 불러온 이미지를 HSV 색공간에서 분석하여 히스토그램을 출력합니다.
- 원하는 색조 값을 입력 받아 이미지의 전체적인 색조를 변경합니다.

제공된 images.zip 파일에 대한 설명은 13 쪽 참고 사항의 ‘1. 실행 결과 예시’ 부분을 참고하세요.

(프로그램 실행 및 이미지 불러오기)
프로그램을 실행하면, 아래와 같이 출력 후 사용자로부터 이미지 파일 이름을 입력 받기 위하여 기다립니다.

```
Enter input filename:
```

Figure 2 프로그램 실행 초기 화면

이미지가 유효한 경우 이미지 파일로부터 데이터를 읽어 이미지의 정보를 각 색공간에 맞는 배열에 저장합니다(RGB, HSV 색공간). 이미지를 불러오는데 성공을 하면 선택 메뉴와 함께 불러온 이미지 파일 명을 함께 출력합니다.
(예시의 빨간색 밑줄은 사용자 입력에 해당)

```
Enter input filename: slime.ppm
====================
  IMAGE COLOR CHANGER
====================
  1. Image Histogram
  2. Change Color
  3. Print Image
  4. Save Image
  5. Exit
Loaded file: slime.ppm

Choose menu number >>
```

Figure 3 파일을 성공적으로 불러온 경우

이미지 파일의 위치는 프로그램이 실행되고 있는 폴더로, Visual Studio 와 Dev-C++ 에디터에서 프로그램 실행 시 별다른 설정이 없을 경우 소스코드(assn2.c)와 같은 폴더에 위치합니다. 또한 파일의 이름은 공백을 포함하지 않으며, 파일 이름의 최대 길이는 74 로 제한합니다. 이미지 배열의 너비, 높이의 최대 값 또한 75 로 제한합니다.

*** 에러 처리**: 파일이 존재하지 않을 경우 “File not found: [파일 이름]”을 출력 후, 프로그램을 종료합니다.

```
Enter input filename: slime11.ppm
File not found: slime11.ppm

계속하려면 아무 키나 누르십시오...
```

Figure 4 파일이 존재하지 않은 경우

(메뉴 화면)

프로그램 실행 후, 이미지를 성공적으로 불러오면 아래와 같이 선택 메뉴를 출력 하고, 사용자의 입력을 받아 해당 기능을 수행합니다. 하나의 기능을 완료하면 다시 메뉴가 출력되도록 합니다.

```
Enter input filename: slime.ppm
======================
  IMAGE COLOR CHANGER
======================
  1. Image Histogram
  2. Change Color
  3. Print Image
  4. Save Image
  5. Exit
Loaded file: slime.ppm

Choose menu number >> 6
Wrong input!
Choose menu number >>
```

Figure 5 메뉴 화면 출력 및 오류 예시

프로그램 화면 출력의 경우 띄어쓰기, '='의 개수 등의 사소한 형식 차이는 감점되지 않습니다.

*** 에러 처리**: 존재하지 않는 메뉴를 선택한 경우 "Wrong input!"을 출력합니다.

(이미지 출력)

메뉴 3번을 선택한 경우, 불러온 이미지를 출력합니다. 이 때, 이미지 출력은 위에서 설명한 RGB 색공간과 콘솔에 픽셀 색을 출력하는 함수를 활용하여 구현합니다. 아래의 이미지는 함께 제공된 ppm 파일을 출력한 예시 입니다.

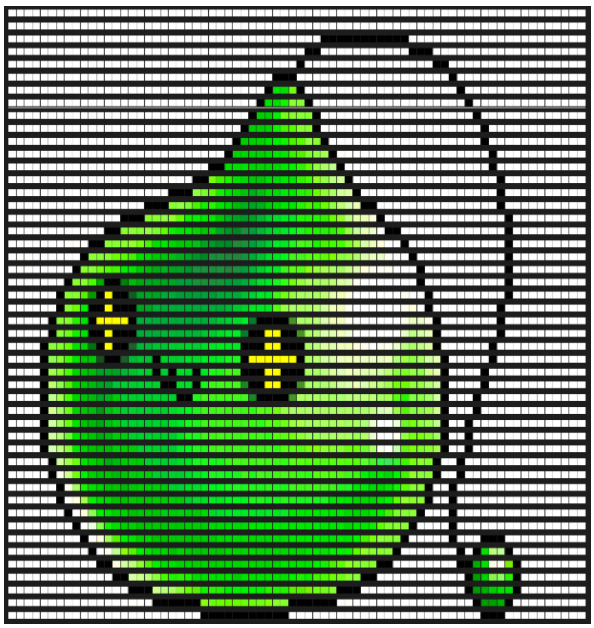


Figure 6 슬라임 이미지 출력

Figure 7 주황버섯 이미지 출력

(이미지 히스토그램 출력)

메뉴 1 번을 선택한 경우, 불러온 이미지에 대하여 HSV 색공간 중 H 값을 분석하여 히스토그램을 출력합니다. 앞서 설명한 것과 같이 H 는 색조를 의미하며, 0<=H<360 사이의 값을 가집니다. 이번 과제에서는 색조의 값을 30 으로 나눈 12 개의 몫을 계급으로 설정한 히스토그램을 그립니다.

자세한 사항은 아래의 이미지를 참조하여 구현합니다. 각 구간의 픽셀 색깔은 해당 index 에 30 을 곱한 값을 색조로, 채도와 명도는 1 을 갖는 색깔로 설정합니다. 즉, 예시로 2 번 계급의 색깔은 H=2*30=60, S=1, V=1 인 색깔입니다.

이 때, S(채도)가 0 인 값은 무채색이기 때문에, 히스토그램을 계산할 때 빈도수로 계산하지않습니다. 또한, 색깔을 출력할 때는 RGB 색공간의 값으로 바꾸어야 한다는 것을 유의해야 합니다. 추가적으로 픽셀의 개수는 빈도 수만큼 나타내기에는 너무 많기 때문에 해당 값의 1/10 만 출력합니다.

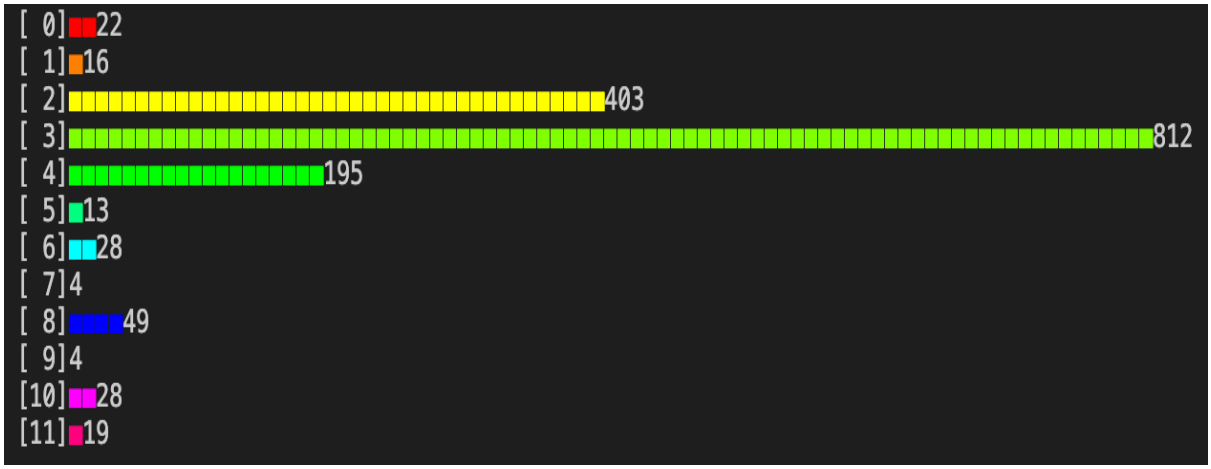


Figure 8 슬라임의 이미지 히스토그램

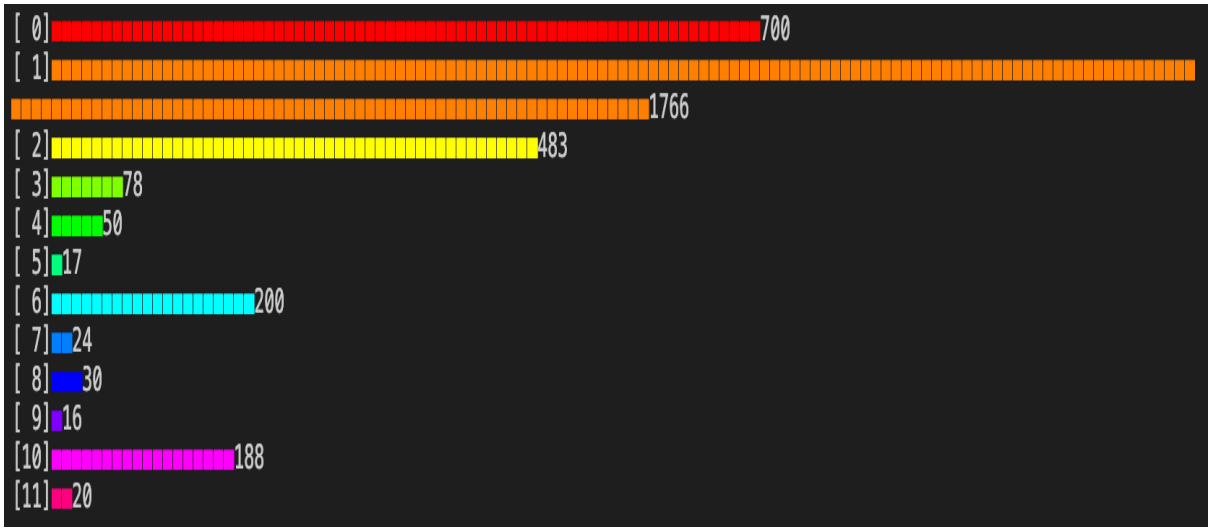


Figure 9 주황버섯의 이미지 히스토그램

※구현 시, 히스토그램 빈도수가 차이가 많이 나는 경우 15 쪽 참고 사항의 '2. *Histogram 출력 시 float 연산 오차*' 부분을 참고하세요.

(이미지 색조 변환)

메뉴 2 번을 선택한 경우, 이미지 히스토그램을 출력한 후, source color 와 target color 를 받아서 이미지의 색조를 변환합니다(HSV 색공간). 이 때, source color 는 바꿀 기준이 되는 색조이며, 이미지는 target color 의 색조로 바뀌게 됩니다. 색조는 이미지의 H 값을 바꾸어 변경할 수 있습니다. 예를 들어 3 번 색조에서 8 번 색조로 바꾸는 경우에는 이미지 각 픽셀의 H 값에 150 을 더하면 됩니다.

색조 변환 시, 변환된 값이 0<=H<360 사이의 값을 가지도록 해야 합니다. 즉, H 가 음수값을 가지면 360 을 더하고 360 이상의 값을 가지면 360 을 빼서 해당 범위를 유지하도록 해야 합니다.

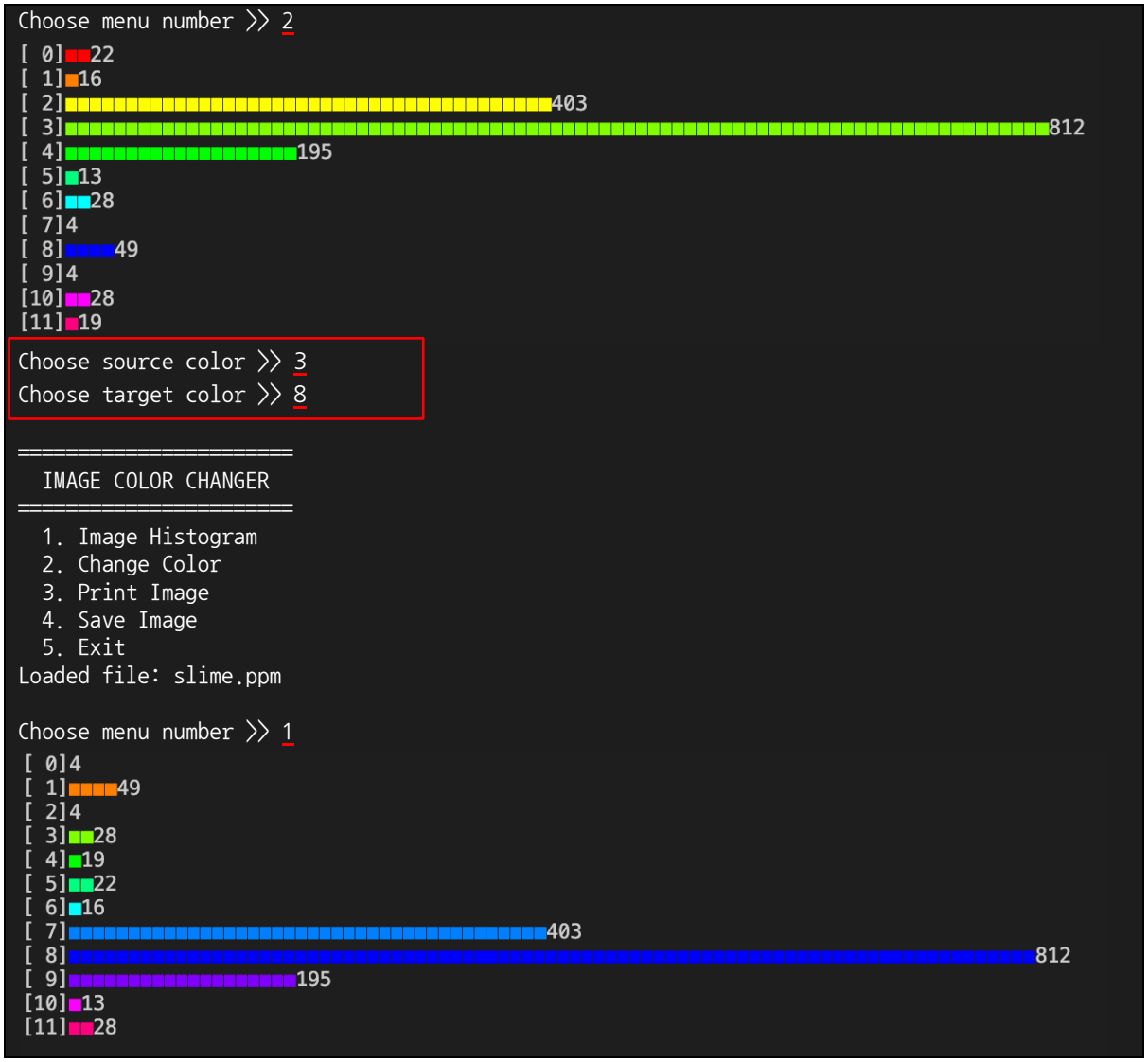


Figure 10 슬라임의 이미지 색조 변환 예시(3번 색조에서 8번 색조로 변환)

슬라임의 색조 변환 후, 메뉴 3번을 선택하여 이미지(RGB 색공간)를 출력하면 <Figure 12>와 같이 색조가 반영된 이미지를 볼 수 있습니다. 색조 변환 시, 색조가 반영된 HSV 색공간 배열 정보로 RGB 색공간 배열 정보를 업데이트해야 합니다.

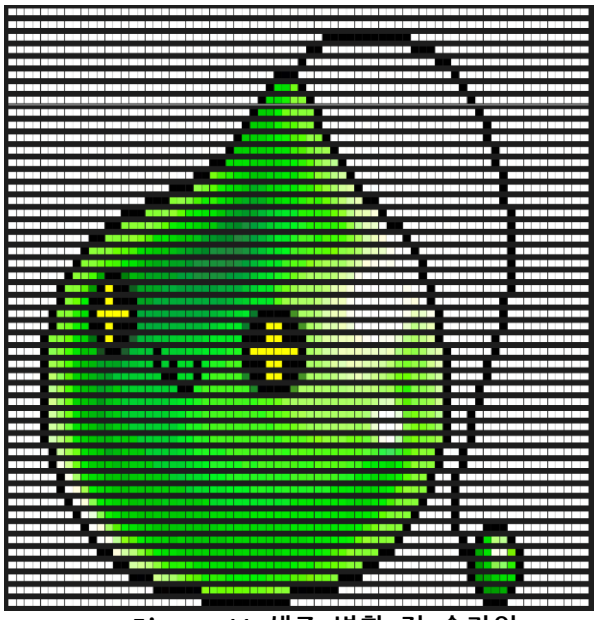


Figure 11 색조 변환 전 슬라임

Figure 12 색조 변환 후 슬라임

*** 에러 처리:** 각 색조는 0 부터 11 까지의 값을 가질 수 있으며, 존재하지 않는 색조를 선택한 경우 "Wrong input!"을 출력 후, 아래 예시처럼 적절한 값을 입력할 때까지 다시 입력 받습니다.

```
Choose source color >> 20
Wrong input!
Choose source color >> 3
Choose target color >> 64
Wrong input!
Choose target color >> 2
```

Figure 13 색조 변환 값 입력 오류 예시

(이미지 저장)

메뉴 4번을 선택한 경우, 색조를 변경한 이미지를 위에서 설명한 ppm 파일 형식으로 저장합니다. 이 때, 저장하는 파일 이름은 "output.ppm"으로 지정합니다. 저장한 ppm 파일을 다시 불러오더라도 정상적으로 출력이 가능해야 합니다.

(필수 구현 함수)

프로그램을 작성할 때 다음 함수들을 반드시 작성하여야 합니다. 그 외에 필요한 함수가 있다면 자유롭게 정의할 수 있습니다. 아래 명시된 함수 이름과 기능은 변경하지 말되, 매개 변수의 개수와 자료형, 반환 자료형 등은 자유롭게 변경이 가능합니다. 이 때 무엇을 어떻게 변경해서 구현했는지 보고서에 기록하도록 합니다. 이미지와 파일 이름의 최대 크기는 75 이며, #define 지시자를 이용하여 크기를 지정하여 사용합니다.

```
#define SIZE 75

int load_image(const char *filename, int image_rgb[][SIZE][SIZE],
                               float image_hsv[][SIZE][SIZE], int *width, int *height);
    -   입력 받은 파일 이름을 통해 이미지를 읽고 이미지의 정보를 각 색공간에 맞는 배열에
        저장합니다.
    -   또한, 이미지의 너비와 높이를 전달받은 포인터에 알맞게 전달합니다.

void save_image(int image_rgb[][SIZE][SIZE], int width, int height);
    -   전달받은 이미지를 ppm 형식의 파일로 저장하는 함수 입니다.
```

```
void print_image(int image_rgb[][SIZE][SIZE], int width, int height);
    -   전달받은 이미지를 출력하는 함수 입니다.

void print_histogram(float image_hsv[][SIZE][SIZE], int width, int height);
    -   전달받은 이미지의 hsv 값을 통해 이미지 히스토그램을 출력하는 함수입니다.

void change_color(float image_hsv[][SIZE][SIZE], int width, int height, int source, int
target);
    -   전달받은 이미지의 hsv 값을 변경하여 이미지의 색조를 변경하는 함수입니다.
```

(참고 사항)

1. 실행 결과 예시

제공된 압축 파일에 입력 이미지 외에도 실행 결과 이미지도 함께 포함되어 있습니다. 각 파일의 이름은 “ 이미지이름_source_target.ppm ” 으로 저장되어 있으며, 이미지 출력 결과는 각각 아래와 같습니다.
예를 들면, 파일 이름 ribbon_0_5.ppm 은 ribbon.ppm 이미지를 0 번 색조에서 5 번 색조로 변경 후 저장한 파일입니다.

(1) ribbon.ppm

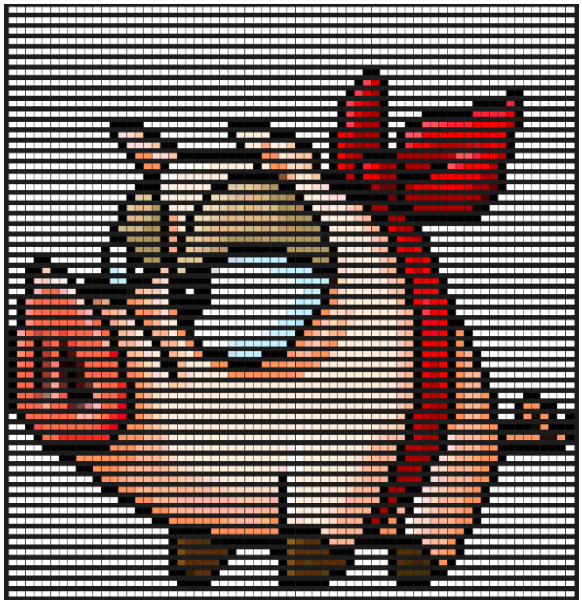


Figure 14 ribbon.ppm

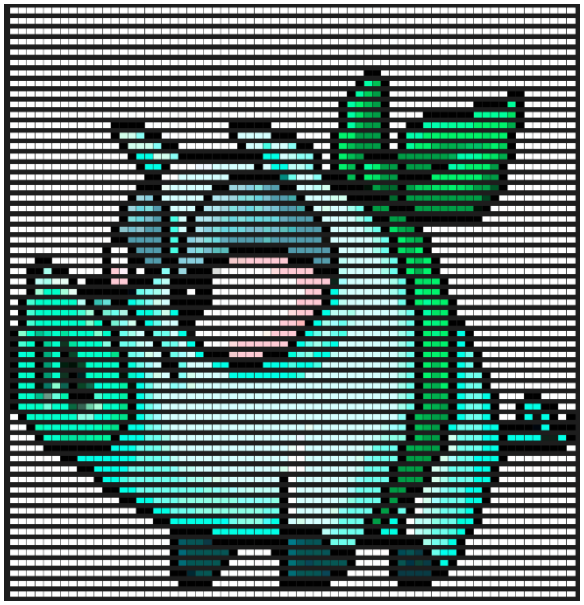


Figure 15 ribbon_0_5.ppm

(2) slime.ppm

Figure 16 slime.ppm

Figure 17 slime_3_8.ppm

(3) gem.ppm

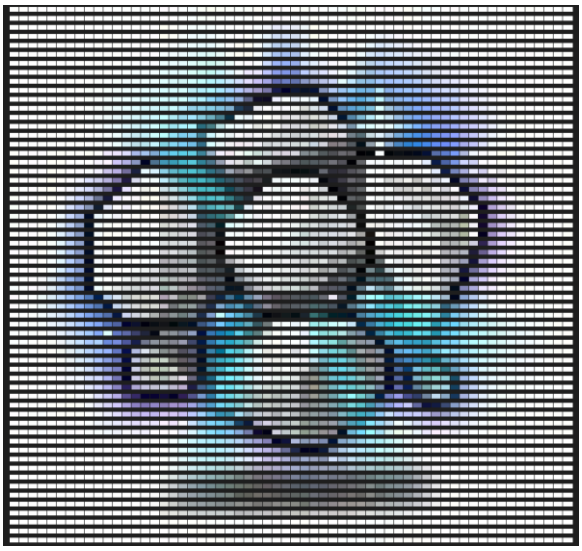


Figure 18 gem.ppm

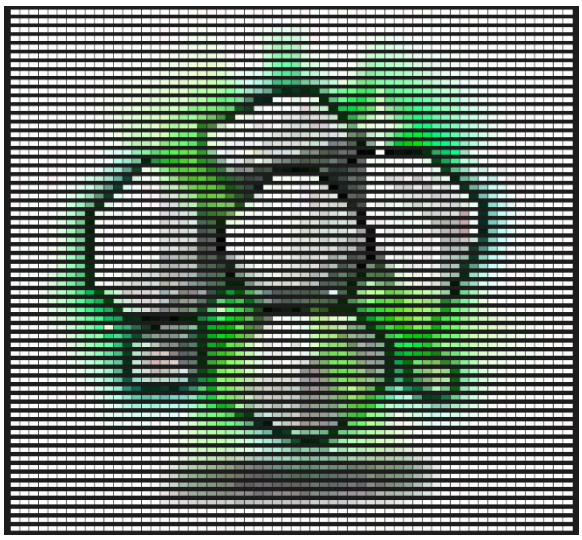


Figure 19 gem_6_3.ppm

(4) mushroom.ppm

Figure 20 mushroom.ppm.

Figure 21 mushroom_1_9.ppm

2. Histogram 출력 시 float 연산 오차

Float 는 더 많은 범위를 표현하기 위해 부동 소수점 방식으로 숫자를 표현합니다. 하지만, 해당 방법으로 숫자를 표현하면 연산을 한 후에는 항상 오차가 존재한다는 단점을 가지게 됩니다. 아래의 예시로 설명하면, 0.1 을 10 번 더한 숫자는 1 이 되어야 하지만, 실제 값은 1.00000119 가 됩니다.

```
#include <stdio.h>

int main()
{
    float x = 0;
    int i = 0;

    for (i = 0 ; i < 10 ; i++){
        x += 0.1;
    }

    if (x == 1) {
        printf("x 는 1 과 같습니다.\n");
    }
    else if (x > 1){
        printf("x 는 1 보다 큽니다.\n");
    }
    else {
        printf("x 는 1 과 작습니다.\n");
    }

    printf("%.9f\n", x);

    return 0;
}
```

```
x 는 1 보다 큽니다.
1.00000119

계속하려면 아무 키나 누르십시오...
```

Histogram 을 출력할 때, Hue 의 값을 비교하여 계급을 정하게 되는데, 이 때 경계값에 있는 값은 오차로 인해 과제의 예시와는 다른 계급에 속할 수 있게 됩니다. 예를 들어 연산 결과 값이 60 이어야 오차로 인해 59.999995 로 나오는 경우가 있을 수 있습니다. 이런 경우에는 계급 [2]에 속해야 하지만 [1]에 속하게 되는 문제점이 발생하게 됩니다. 이러한 문제점은 연산값의 결과가 오차 내에 있는지 확인함으로써 해결할 수 있습니다. 아래의 코드를 보면 x 와 비교할 값인 1 의 값 차이가 float 의 오차인 0.00001 보다 작으면 같은 값으로 처리하는 것을 알 수 있습니다. fabsf() 함수는 <math.h> 내부에 있는 함수로써, float 의 절댓값을 계산해주는 함수 입니다. 이번 과제에서는 ERROR 를 0.00001 로 정의하여 사용합니다.

```
#include <stdio.h>
#include <math.h>
#define ERROR 0.00001 // float 의 오차

int main()
{
    float x = 0;
    int i = 0;
```

```
    for (i = 0; i < 10; i++){
        x += 0.1;
    }

    if (fabsf(x - 1) < ERROR) {
        printf("x는 1과 같습니다.\n");
    }
    else if (x > 1){
        printf("x는 1보다 큽니다.\n");
    }
    else {
        printf("x는 1과 작습니다.\n");
    }
    return 0;
}
```

```
x는 1과 같습니다.

계속하려면 아무 키나 누르십시오...
```