CSED101. Programming & Problem solving

Fall, 2018

Programming Assignment #4

(70 points)

이성현(sh0416@postech.ac.kr)

- ***Due***: **2018.11.29 23:59**
- ***Development Environment:*** **Windows Visual Studio 2017**

- ***제출물***
  - **C Code 압축 파일 (assn4.zip)**
    - 파일목록: *mystring.h, mystring.c, assn4.c, image.h, image.c*
      - *image.h, image.c*: 이미지 구조체 타입 정의 및 이미지 처리 관련 함수
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
  - **보고서 파일** (assn4.docx 또는 assn4.hwp)
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - **명예서약(Honor code)**: 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
    - 보고서는 Problem2에 대해서만 작성할 것
    - 보고서에는 'girl.ppm'과 'leaf.ppm'을 흐릿함 처리한 결과와 테두리를 검출한 결과가 포함되어야 한다.
  - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- ***주의사항***
  - 각 문제에 해당하는 요구사항을 반드시 지킬 것.
  - 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
  - 각 문제에 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다.
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
  - 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem 1: 문자열 처리 함수 (5점)

[문제)

C 언어에서 제공되는 문자열 함수는 "string.h" 라이브러리에서 제공된다. 문자열 라이브러리에 포함되어 있는 함수 중 일부를 직접 작성해 보자.

(설명)

제공된 assn4_p1.zip 의 압축을 풀면 **mystring.h** 와 **mystring.c** 가 주어진다.

mystring.h 는 아래와 같은 문자열 처리 함수의 선언을 포함하고 있다. (그대로 사용하고 변경하지 말 것)

```
#ifndef MYSTRING_H
#define MYSTRING_H

int mystrlen(char *str);
char *mystrcpy(char *toStr, char *fromStr);
char *mystrcat(char *str1, char *str2);
char *mystrchr(char *str, char c);

#endif
```

위 함수의 구현부를 포함한 mystring.c 를 작성하라. 각 함수의 정의는 다음과 같다.

(1) int mystrlen(char *str)

NULL 문자('₩0')를 제외한 문자열의 길이를 반환한다. 빈 문자열의 경우 0 을 반환한다.

예제)

```
printf("%d\n", mystrlen("cs101")); // 결과: 5
```

(2) char *mystrcpy(char *toStr, char *fromStr)

문자열 복사 함수로 NULL 문자를 포함한 문자열 fromStr 를 문자열 toStr 에 복사한 후, 문자열 toStr 의 시작 주소를 반환한다. (단, toStr 에 충분한 저장공간이 있다고 가정)

예제)

```
char str[256];
printf("%s\n", mystrcpy(str, "Good Day")); // 결과: Good Day
printf("%s\n", mystrcpy(str, "Hello"));    // 결과: Hello
```

(3) char *mystrcat(char *str1, char *str2)

문자열 연결함수로 str1 의 끝에 str2 를 이어 붙인다. 즉, 문자열 str1 뒤의 NULL 문자는 str2 의 첫 번째 문자로 덮어 씌워지고 str2 의 NULL 문자는 남는다. 문자열 연결 후, 문자열 str1 의 시작 주소를 반환한다. (단, str1에 충분한 저장공간이 있다고 가정)

예제)

```
char str[256] = "Hello";
mystrcat(str, ", World");
printf("%s\n", str); // 결과: Hello, World
```

(4) char* mystrchr (char *str, char c)

문자열 str에서 가장 처음으로 c와 일치하는 문자c의 시작 주소를 반환한다. 만약 문자열 str에서 일치하는 것이 없으면 NULL을 반환한다.

예제)

```
printf("%s\n", mystrchr("cs101", '1')); // 결과: 101
```

■ Problem 2: 이미지 처리 프로그램 만들기 (65점)

(소개)

이미지에 단순한 필터를 적용하여 이미지에 효과를 부여하는 것은 여러 어플리케이션에 적용되어 있다. 이는 컴퓨터가 단순한 작업을 빠르게 반복할 수 있기 때문에 가능하다. 우리는 이 과제에서 이미지 처리 기능 중에서 가장 간단하게 적용할 수 있는 흐릿함 효과와 이미지 내부에서 물체의 테두리를 검출할 수 있는 기능을 제공할 수 있는 프로그램을 작성하도록 해보자.

두 기능은 **컨볼루션**이라는 이미지 연산 방법으로 처리 가능하다. 컨볼루션은 이미지와 필터를 입력으로 받아서 필터를 이미지에 적용하는 연산을 의미한다. 즉, 흐릿함 효과는 흐릿함 효과를 줄 수 있는 필터를 이용하여 컨볼루션 연산을 진행하면 이미지에 흐릿함 효과가 적용되고, 테두리 검출은 테두리를 검출할 수 있는 필터를 이용하여 컨볼루션 연산을 진행하면 테두리를 검출한 이미지가 만들어지게 된다.

아래 이미지는 이 과제를 모두 마쳤을 때, 제공하고 있는 필터와 이미지를 이용하여 흐릿함 효과를 적용한 결과와 테두리를 검출했을 때의 결과를 보여준다.

아래는 아무런 효과도 주지 않은 기본 이미지이다.

Figure 1 cat.ppm

이미지에 흐릿함 효과를 줄 경우, 이미지 정규화 없이 바로 'blur.ppm'이라는 이름의 필터를 불러들인 후 컨볼루션을 진행하고 그 결과 이미지를 저장하면 흐릿한 이미지가 만들어진다.

Figure 2 cat_blur.ppm

이미지에서 테두리를 검출하려고 할 경우, 이미지 정규화가 필요하다. 이미지 정규화를 한 뒤 'edge.ppm'이라는 이름의 필터를 불러들인 후 컨볼루션을 진행한다. 그리고, 그 결과 이미지를 역정규화를 한 뒤 이미지를 저장하면 테두리 부분을 검출한 이미지가 만들어진다.

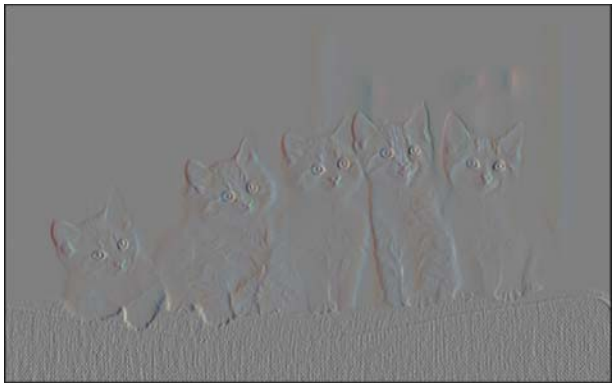


Figure 3 cat_edge.ppm

이 외에도 다른 필터를 이용하여 컨볼루션 연산을 진행할 수도 있다.
제공되는 이미지는 'cat.ppm', 'girl.ppm', 'leaf.ppm'이며 제공되는 필터는 'blur.ppm', 'edge.ppm'이다.

(목표)

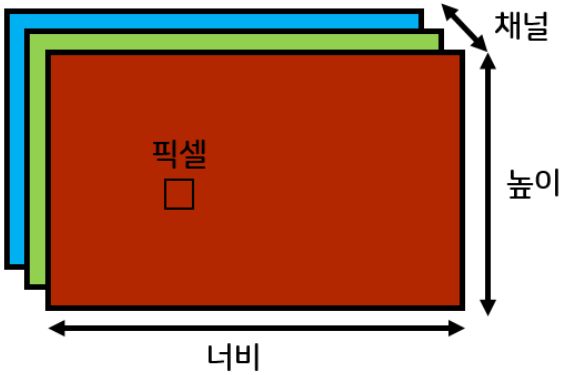
이번 과제의 목표는 다음과 같다.

- 파일 입출력을 이용하여 파일을 읽어 들이고 저장할 수 있다.
- 이미지와 필터를 동적 할당을 이용하여 다차원의 형태로 저장할 수 있다.
- 다차원으로 표현된 데이터를 처리할 수 있다.

이미지를 불러들이고, 저장하고 처리할 수 있는 프로그램을 작성하라.

(이미지)

이미지는 3차원 구조로 만들어져 있다. 3차원 구조에서 각각의 축은 채널(C), 높이(H), 너비(W)로 표현된다. 만약, 채널이 3, 높이가 100, 너비가 200인 이미지가 있다고 했을 때, 이 이미지를 표현하기 위해 필요한 데이터의 개수는 총 3*100*200=60000개가 된다. 각각의 데이터는 0부터 255의 정수를 가지게 된다. 일반적으로 이미지는 빨간색, 녹색, 파란색을 이용하여 색을 표현하고 각각을 하나의 채널로 표현하기 때문에 이미지는 3개의 채널을 가지고 있다.

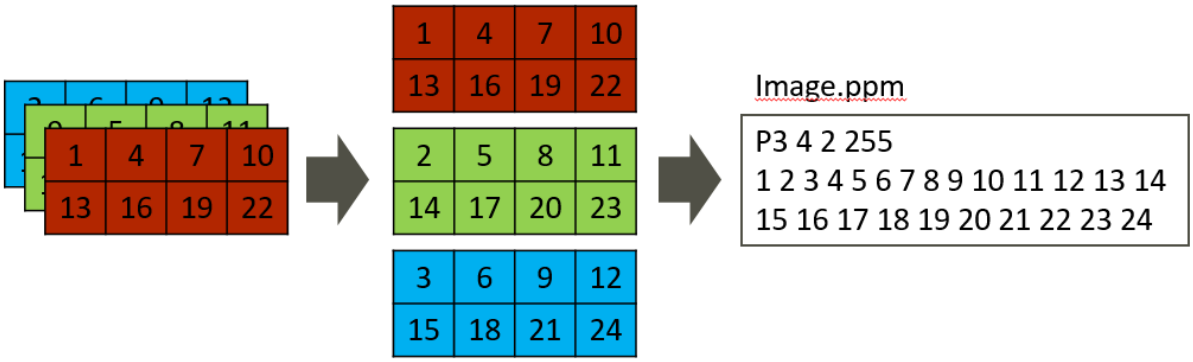


(ppm 파일 포맷)

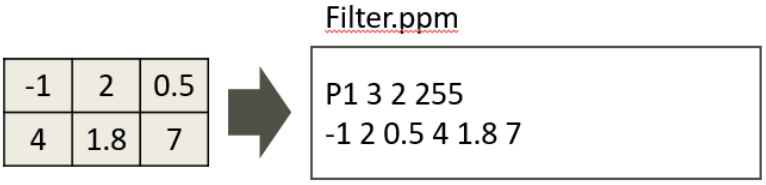
이번 과제에서 사용할 이미지 파일 포맷은 "ppm" 포맷이다. 첫 번째 줄은 이미지의 크기에 관련된 정보를 담고 있다.

```
P3 600 370 255
```

P3은 하나의 픽셀이 3개의 데이터로 이루어져 있다는 의미이며, 600은 이미지의 너비가 600 픽셀이고 높이가 370 픽셀 임을 의미한다. 255는 각각의 데이터가 0부터 255까지의 정수형을 표현할 수 있음을 말한다. 우리가 처리할 이미지는 항상 P3으로 시작하며 255로 끝난다. 높이와 너비는 다를 수 있다.
두 번째 줄은 각각의 픽셀에 대한 데이터가 들어 있다. 나열되어 있는 방법은 채널, 너비, 높이의 순서로 나열되어 있다. 예를 들어, 높이가 2, 너비가 4, 채널이 3인 이미지의 경우 아래와 같은 파일 구조를 가진다.



이번 과제에서 사용할 필터 파일 또한 두 줄로 이루어져 있다. **필터는 채널이 1**이라고 가정하고 각각의 **데이터는 음수도 실수도 저장할 수 있다**고 가정한다. 즉, 우리가 처리할 필터는 항상 P1으로 시작하며 높이와 너비는 변할 수 있고 4번째 값은 의미 없는 값이 된다. 그 이외에는 이미지와 동일한 형태이며 파일 형식 또한 모두 동일하다.



Ppm 파일 포맷을 이미지로 열기 위해서는 꿀뷰 이미지 뷰어가 필요하다. 이는 https://kr.bandisoft.com/honeyview/ 에서 다운로드 받을 수 있으며 ppm 파일을 클릭한 뒤 꿀뷰로 ppm파일을 열게 되면 이미지를 볼 수 있다.

(Normalization 연산)

이번 과제에서 만들어내야 하는 이미지 처리 연산은 중 하나는 정규화이다. 이는 이미지의 데이터가 0부터 255사이에 있기 때문에 이를 -1부터 1사이의 값으로 변환하는 과정을 의미한다. 수학적인 표현으로 나타내면 다음과 같다.

$$\text{Result}_{\text{height}} = Image_{height}$$

$$\text{Result}_{\text{width}} = Image_{width}$$

$$\text{Result}_{\text{channel}} = Image_{channel}$$

$$\text{Result}_{\text{pixel}}[\text{c}][\text{h}][\text{w}] = \frac{\text{Image}_{\text{pixel}}[c][h][w]}{128} - 1$$

이렇게 정규화된 이미지를 다시 0부터 255사이의 값으로 변환하는 과정도 이미지를 포맷에 맞게 저장하기 위해 필요하며 이를 역정규화라고 부른다. 이를 수학적인 표현으로 나타내면 다음과 같다.

$$\text{Result}_{\text{pixel}}[c][h][w] = 128 \times (Image_{pixel}[c][h][w] + 1)$$

(Convolution 연산)

이번 과제에서 만들어내야 하는 이미지 처리 연산은 컨볼루션(Convolution)이다. 이는 필터를 이용하여 이미지를 처리하는 기법 중 하나이다.

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html 의 첫 번째 그림이 이번 과제에서 만들어야 할 컨볼루션 연산이다. 위 링크에서는 이미지가 하나의 채널에 대해서만 적용되지만 **우리 과제에서는 채널이 3개이므로 각각의 채널에 대해서 같은 필터로 컨볼루션 연산을 진행한다.** 어떤 이미지와 필터가 주어졌을 때, 그것의 컨볼루션 결과값은 실제 이미지보다 조금 더 작아지며 그 결과값의 크기는 다음과 같다.

$$\text{Result}_{\text{height}} = Image_{height} - Filter_{height} + 1$$

$$\text{Result}_{\text{width}} = Image_{width} - Filter_{width} + 1$$

$$\text{Result}_{\text{channel}} = Image_{channel}$$

결과값의 픽셀을 계산하는 방법은 다음과 같다.

$$\text{Result}_{\text{pixel}}[c][rh][rw] = \sum_{h=0}^{Filter_{height}} \sum_{w=0}^{Filter_{width}} Image_{pixel}[c][rh+h][rw+w] \times Filter_{pixel}[h][w]$$

이미지가 필터보다 작을 경우, 결과 이미지의 크기가 음수가 될 수도 있는데 이런 경우는 고려하지 않는 것으로 한다.

(요구사항)

이번 과제에서는 여러 개의 이미지와 여러 개의 필터를 받아들이고 이미지와 필터 간에 연산을 할 수 있는 프로그램을 구현하는 것이다. 이미지는 아래에 주어진 자료형을 기반으로 저장해야 하며 여러 개의 이미지와 필터를 저장하기 위해 동적 할당을 이용해야 한다.

```
typedef struct {
    char name[100];    // File name
    float*** data;     // Pixel for CUBIC data
    int C, H, W;       // C: channel, H: height, W: width
    int is_normalized; // Whether image is normalized or not
}CUBIC;
```

여러 개의 이미지와 필터는 동적 할당을 이용하여 CUBIC 포인터 배열을 생성해야 하며 매 순간 필요하지 않은 공간이 없도록 해야 한다. 즉, 처음에는 포인터 배열을 위한 동적 할당이 일어나지 않다가 이미지를 저장할 공간이 필요해졌을 때, 포인터 배열을 한 칸 늘려 저장할 공간을 확보해야 한다. 또한, 이미지를 삭제하는 기능을 진행할 때에도 마찬가지로 이미지를 삭제한 뒤, 포인터 배열의 크기를 한 칸 지워서 압축된 형태로 관리해야 한다.

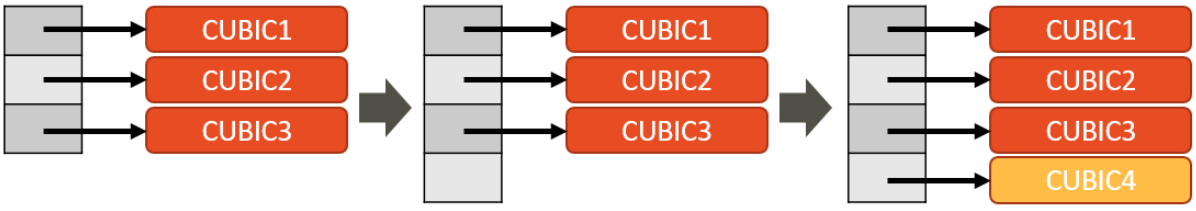


Figure 4 이미지를 추가할 때

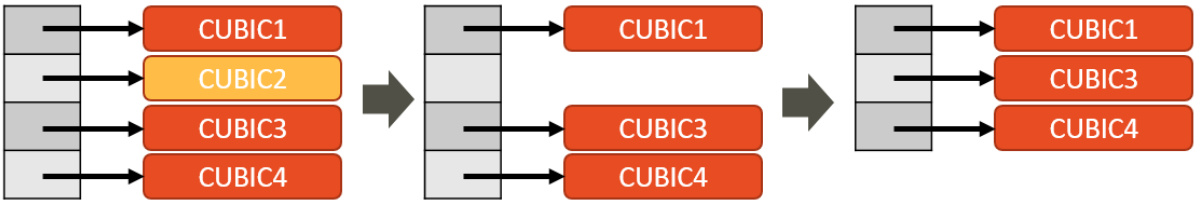


Figure 5 이미지를 삭제할 때 (2번 이미지를 삭제하는 상황)

(주의사항)

1. 출력은 실행 예시들과 최대한 동일하게 하되, 띄어쓰기나 "="의 개수는 달라도 무관하다. (모든 공백은 'space'를 사용한다.)
2. 표준 헤더 파일 <string.h>를 include하여 사용할 수 있다.
3. 과제 작성시 전역변수는 사용할 수 없으며, 각 기능을 수행하는 함수를 별도로 구현할 것.
4. 이미지 처리와 관련된다고 생각하는 함수들은 "image.c" 파일에 구현할 것 (반드시 하나 이상의 함수가 "image.c"에 들어있어야 한다.
5. 프로그램 시작 시, 아래와 같이 main()함수에서 이미지 목록 포인터와 필터 목록 포인터로만 시작하도록 한다.

```
CUBIC **images = NULL;
CUBIC **filters = NULL;
```

   이미지 또는 필터 저장 시에 동적 할당을 이용해 데이터를 저장하고, 삭제 시에 free를 이용하여 할당 해제 하도록 한다.
6. 연결 리스트(linked list)와 같은 다른 방법을 사용할 경우, 감점 처리한다.
7. 프로그램이 수행 될 때, 발생할 수 있는 예외 상황들을 고려하여 구현한다. 예외 상황 발생시, 예외 상황임을 알릴 수 있는 적절한 문장을 출력하도록 한다. (아래 명시하지 않은 예외 상황에 대해서는 고려할 필요가 없다.)
   1) 유효하지 않은 범위 값을 입력할 경우
      - 메뉴 번호 입력 시
      - 이미지 및 필터 목록에서 인덱스를 선택하여 입력 시
   2) 이미지 및 필터 불러오기에서 읽을 파일이 없는 경우
      단, 이미지 또는 필터 파일의 형식이 다르거나, 데이터 개수가 다른 경우에 대해서는 고려할 필요가 없다.
   3) 정규화된 이미지를 정규화하려고 할 때, 역정규화된 이미지를 역정규화하려고 하는 경우

(구현 기능 설명)

1. **메뉴 출력 및 입력**

   프로그램이 실행 되면 아래의 메뉴가 출력되며, 하나의 기능을 완료하면 메뉴는 다시 출력되도록 한다. (메뉴명의 띄어쓰기나 "="의 개수가 달라도 감점되지 않는다.)

   메뉴가 출력 된 뒤에는 사용자 입력을 받는다.

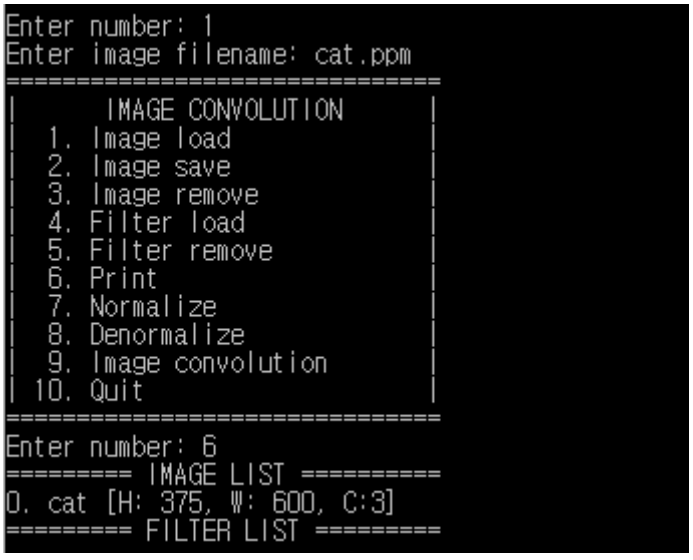
2. **이미지 불러들이기**

   i. '1'을 입력하여 '***Image load***'를 선택할 경우, 아래와 같이 'Enter image filename: '이라는 메시지 출력 후, 사용자로부터 이미지 파일 이름을 입력 받는다.

   ii. 이미지가 유효한 경우, 이미지 파일로부터 데이터를 읽어 들여 주어진 자료형에 맞게 저장한다. (반드시 주어진 자료형을 사용하여야 한다.)

   iii. 'name'은 확장자 명(.ppm)을 제외한 파일 이름으로 초기화한다. 확장자 명을 제외한 파일 이름은 띄어쓰기 없이 영어, 숫자, 기호(_)의 조합으로 이루어져 있다고 가정한다.

   iv. 이미지 배열에 추가한다. (Figure 4처럼 배열의 맨 아래에 이미지를 추가한다.)



3. **이미지 및 필터 목록 출력하기**

   i. '6'을 입력하여 '***Print***'를 선택할 경우, 위의 예시처럼 이미지와 필터 목록을 출력한다. 위의 예시는 이미지 목록은 있고 필터 목록은 없는 경우의 예시이다.

   ii. 위의 실행 예시를 기반으로 인덱스, name, C, H, W, is_normalized를 모두 출력한다. is_normalized의 출력 예시는 9쪽의 '*7. 이미지 정규화하기*'의 예시를 참고한다.

4. **이미지 저장하기**

i. '2'을 입력하여 '***Image save***'를 선택할 경우, 아래와 같이 이미지 목록이 출력되며 사용자로부터 저장할 이미지를 인덱스로 입력 받는다.

```
Enter number: 2
0. cat [H: 375, W: 600, C:3]
Enter the number of image to save: 0
Enter filename for save image: cat2.ppm
```

ii. 다음으로, 선택한 이미지를 저장할 이미지 파일 이름을 입력 받는다.

iii. 저장할 이미지를 ppm 파일 형식으로 저장한다.

iv. 이미지를 실수형으로 가지고 있지만 파일에 저장할 땐 정수형으로 변환하여 저장하고 만약 값이 0보다 작으면 0, 255보다 크면 255로 저장한다. 소수점 이하는 버린다.

v. 이미지 불러오기를 통하여 cat.ppm을 불러온 후, 바로 이미지 저장하기를 통하여 cat2.ppm으로 저장한 경우, cat2.ppm은 cat.ppm과 같은 이미지로 생성된다.

5. **필터 불러들이기**

i. '4'를 입력하여 '***Filter load***'를 선택할 경우, 'Enter filter filename: '이라는 메시지 출력 후, 사용자로부터 필터 파일 이름을 입력 받는다.

ii. 필터 파일이 유효한 경우, 필터 파일로부터 데이터를 읽어 들여 주어진 자료형에 맞게 저장한다. (반드시 주어진 자료형을 사용하여야 한다.)

iii. 'name'은 확장자 명(.ppm)을 제외한 파일 이름으로 초기화한다.

iv. 필터 배열에 추가한다. (Figure 4처럼 배열의 맨 아래에 필터를 추가한다.)

v. '6'을 입력하여 추가된 필터 목록을 확인하면 아래와 같다.

```
Enter number: 4
Enter filter filename: blur.ppm
==============================
|       IMAGE CONVOLUTION     |
|  1. Image load              |
|  2. Image save              |
|  3. Image remove            |
|  4. Filter load             |
|  5. Filter remove           |
|  6. Print                   |
|  7. Normalize               |
|  8. Denormalize             |
|  9. Image convolution       |
| 10. Quit                    |
==============================
Enter number: 6
========= IMAGE LIST ==========
0. cat [H: 375, W: 600, C:3]
========= FILTER LIST =========
0. blur [H: 5, W: 5, C:1]
```

6. **이미지 컨볼루션 연산하기**

i. '9'를 입력하여 '***Image convolution***'를 선택할 경우, 아래와 같이 이미지 목록이 출력되며 사용자로부터 컨볼루션 연산을 진행할 이미지를 인덱스로 입력 받는다.

ii. 다음으로, 필터 목록이 출력되며 컨볼루션 연산을 진행할 필터를 인덱스로 입력 받는다.

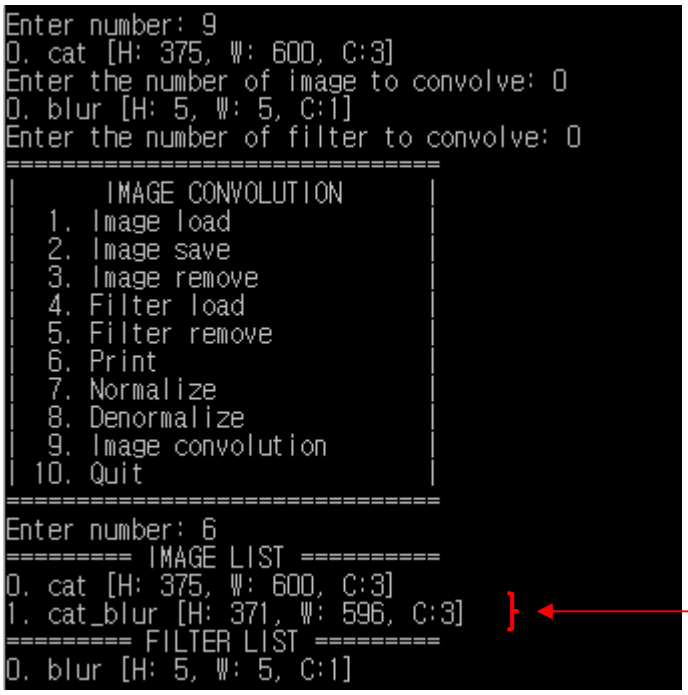
iii. 주어진 이미지와 필터를 가지고 컨볼루션 연산을 진행한 결과값을 생성한다.

iv. 정규화된 이미지로 컨볼루션을 진행했으면, 결과값의 'is_normalized'는 1이어야 하고, 정규화되지 않은 이미지로 컨볼루션을 진행했으면, 결과값의 'is_normalized'는 0이어야 한다.

v. 'name'은 '이미지이름_필터이름'으로 초기화 한다.

vi. 이를 이미지 배열에 추가한다. (배열의 맨 아래에 이미지를 추가한다.)

vii. '6'을 입력하여 추가된 이미지 목록을 확인하면 아래와 같다.



추가된 이미지

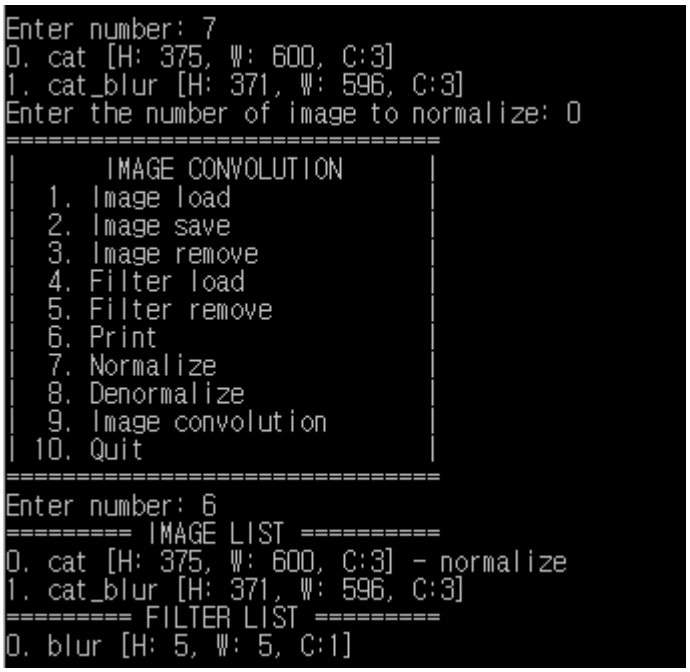
viii. '2'를 입력하여, 컨볼루션한 이미지를 아래와 같이 저장한다.

ix. 꿀뷰를 통하여 cat_blur.ppm을 열어보면 Figure 2와 같이 블러 필터가 적용된 이미지를 확인할 수 있다.

7. **이미지 정규화하기**

   i. '7'을 입력하여 '***Normalize***'를 선택할 경우, 이미지 목록이 출력되며 정규화할 이미지 입력을 인덱스로 입력 받아 정규화 한다.

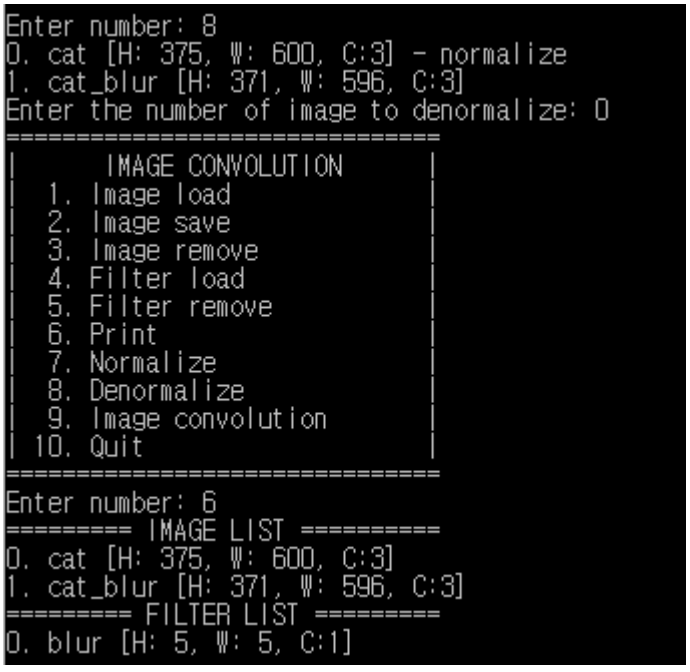
   ii. 정규화 후, 이미지 목록을 출력하면 아래와 같이 출력된다.



8. **이미지 역정규화하기**

   i. '8'를 입력하여 '***Denormalize***'를 선택할 경우, 아래와 같이 이미지 목록이 출력되고, 역정규화할 이미지 입력을 인덱스로 입력 받아 역정규화 한다.

   ii. 역정규화 후, 이미지 목록을 출력하면 아래와 같이 출력된다.

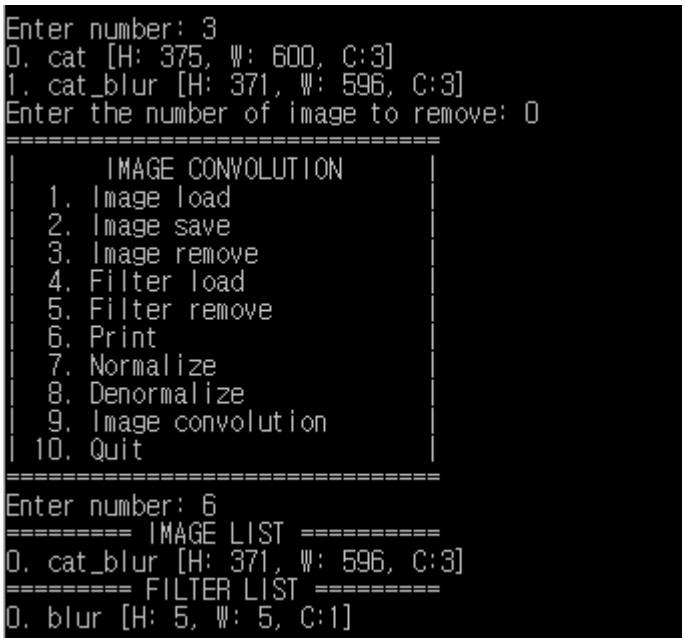


9. **이미지 삭제하기**

i. '3'을 입력하여 '***Image remove***'를 선택할 경우, 아래와 같이 이미지 목록이 출력되며 사용자로부터 삭제할 이미지를 인덱스로 입력 받는다.

ii. 선택한 이미지를 삭제한다. (Figure 5처럼 이미지들을 한 칸씩 위로 끌어올린 뒤 배열을 압축시켜야 한다.) 이때, 이미지 저장을 위해 동적 할당 받은 공간을 할당해제하도록 한다.

iii. '6'을 입력하면 아래처럼 이미지 목록이 삭제되었음을 확인할 수 있다.



10. **필터 삭제하기**

i. '5'를 입력하여 '***Filter remove***'를 선택할 경우, 아래와 같이 필터 목록이 출력되며 사용자로부터 삭제할 필터를 인덱스로 입력 받는다.

ii. 선택한 필터를 삭제한다. (Figure 5처럼 필터들을 한 칸씩 위로 끌어올린 뒤 배열을 압축시켜야 한다.) 이때, 필터 저장을 위해 동적 할당 받은 공간을 할당해제하도록 한다.

iii. '6'을 입력하면 아래처럼 필터 목록이 삭제되었음을 확인할 수 있다.

11. **프로그램 종료**

i. 프로그램을 종료한다.

ii. 모든 이미지와 필터가 올바르게 할당 해제 되어야 한다.