CSED101 Programming & Problem Solving
Fall, 2018
Programming Assignment #4

무은재 새내기학부
20180038
박형규
POVIS ID : hyeongkyu

Honor Code : 나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

1. 프로그램 개요

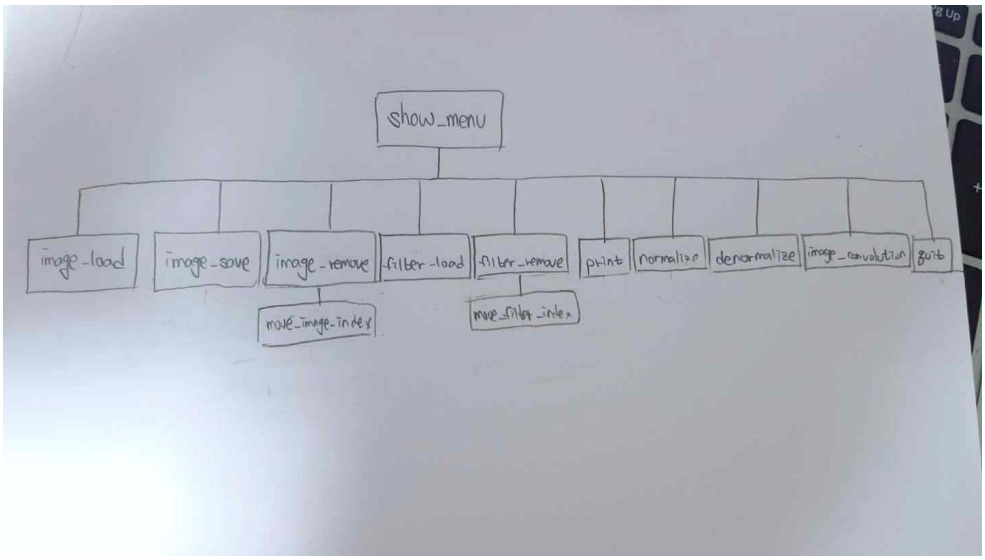
1.1 목적

이번 프로그램을 통하여 이전에 배웠던 파일 입출력을 이용하여 파일을 읽어 들이는 법을 복습하고 이미지와 필터를 동적 할당을 이용하여 다차원의 형태로 저장하는 방법을 익힌다. 또한 다차원으로 표현된 데이터를 처리하는 방법을 익힌다.

1.2 전체 구조

이번 과제는 우리가 흔히 쓰는 이미지에 필터를 넣는 프로그램을 실제로 만들어서 그 과정을 이해하는데 그 목적이 있다. 이미지 파일과 필터 파일을 불러들여서 정규화, 역정규화, 컨볼류젼 연산 등을 처리하여 이미지에 필터를 입혀 그 파일을 출력해 내는 구조이다.

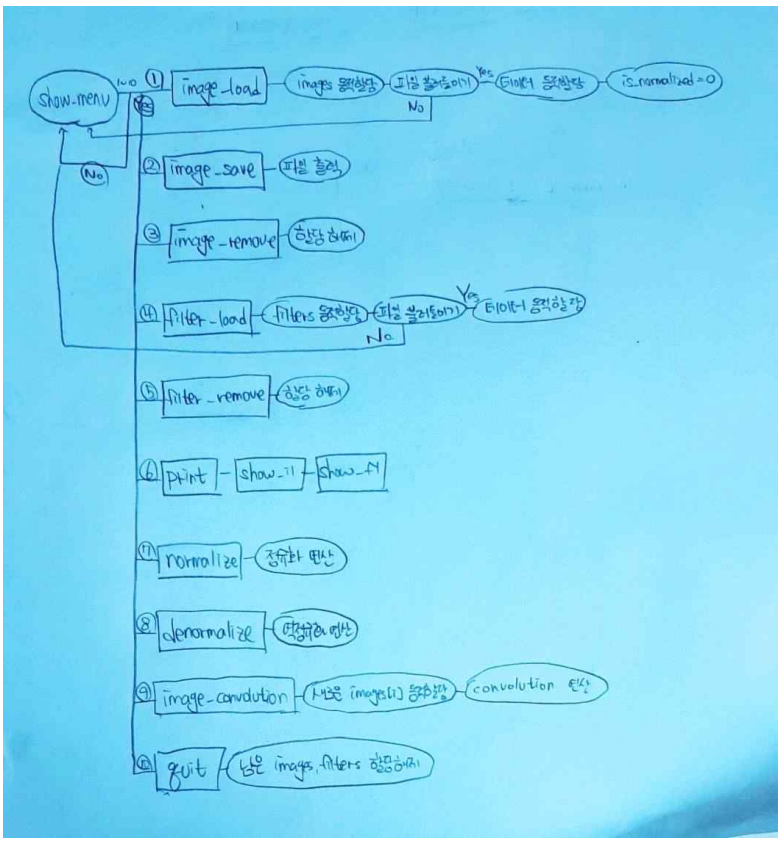
2. 전체 구조도



3. pseudocode

```
Algorithm image & filter
show_menu();
image_load(CUBIC **images, int I) || image_remove(CUBIC **images, int I);
filter_load(CUBIC **filters, int j) || filter_remove(CUBIC **filters, int j;
normalize(CUBIC **images, int I) || denormalize(CUBIC **images, int I);
image_convolution(CUBIC **images, CUBIC **filters, int I, int j);
print(CUBIC **images, CUBIC **filters, int I, int j);
image_save(CUBIC **images, int I);
quit(CUBIC **images, CUBIC **filters, int I, int j);
```

4. flow chart



5. 프로그램 코드

Header file 포함 & CUBIC 구조체 선언

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct CUBIC
{
char name[100];
float*** data;
int C, H, W;
int is_normalized;
};
typedef struct CUBIC CUBIC
```

함수 선언

```
int show_menu();
void show_il(CUBIC **images, int i);
void show_fl(CUBIC **filters, int j);
void image_load(CUBIC **images, int i);
void print(CUBIC **images, CUBIC **filters, int i, int j);
void image_save(CUBIC **images, int i);
void filter_load(CUBIC **filters, int j);
void image_remove(CUBIC **images, int i);
void filter_remove(CUBIC **filters, int j);
void move_image_index(CUBIC **images, int i, int num);
void move_filter_index(CUBIC **filters, int j, int num);
void normalize(CUBIC **images, int i);
void denormalize(CUBIC **images, int j);
void image_convolution(CUBIC **images, CUBIC **filters, int i, int j);
void quit(CUBIC **images, CUBIC **filters, int i, int j);
```

main 함수 images & filters 동적할당 부분

```
if (i == 0)
{
if (num == 1 || num == 9)
{
images = (CUBIC**)malloc(sizeof(CUBIC*) * (i + 1));
}
}
else
{
if (num == 1 || num == 9)
{
images = (CUBIC**)realloc(images, sizeof(CUBIC*) * (i + 1));
}
}
if (j == 0)
{
if (num == 4)
{
filters = (CUBIC**)malloc(sizeof(CUBIC*) * (j + 1));
}
}
else
{
if (num == 4)
{
filters = (CUBIC**)realloc(filters, sizeof(CUBIC*) * (j + 1));
}
}
```

I가 0일 때, num 값으로 1이나 9가 입력되면 images의 한 칸을 소모해야 하므로 images에 CUBIC**만큼의 한 칸을 malloc을 이용하여 할당해준다. 또한 I가 1 이상일 때 num 값으로 1이나 9가 입력되면 계속해서 images의 한 칸을 소모해야 하므로 realloc을 이용하여 images의 크기를 다시 할당해준다. filter도 마찬가지로 j가 0일 때 num값으로 4가 입력되면 filterse의 한 칸을 소모해야 하므로 filters에 CUBIC**만큼의 한 칸을 malloc을 이용하여 할당해준다. 또한 j가 1 이상일 때 num 값으로 4가 입력되면 계속해서 filters의 한 칸을 소모해야 하므로 realloc을 이용하여 filters의 크기를 다시 할당해준다.

main함수 switch-case 부분

```
switch (num)
{
case 1: image_load(images, i);
i++;
break
case 2: image_save(images, i);
break
case 3: image_remove(images, i);
i--;
break
case 4: filter_load(filters, j);
j++;
break
case 5: filter_remove(filters, j);
j--;
break
case 6: print(images, filters, i, j);
break
case 7: normalize(images, i);
break
case 8: denormalize(images, i);
break
case 9: image_convolution(images, filters, i, j);
if (i == 0)
{
break
}
else
{
i++;
break
}
case 10: quit(images, filters, i, j);
break
default: printf("Please enter one of the 1 - 10 number");
return 0;
break
}
```

switch-case문을 이용하여 num값으로 받아들인 수에 알맞은 작동 함수를 실행시킨다.

show_menu()

```
int num;
printf("=====================================₩n");
printf("I          IMAGE CONVOLUTION          I₩n");
printf("I    1. Image load                    I₩n");
printf("I    2. Image save                    I₩n");
printf("I    3. Image remove                  I₩n");
printf("I    4. Filter load                   I₩n");
printf("I    5. Filter remove                 I₩n");
printf("I    6. Print                         I₩n");
printf("I    7. Normalize                     I₩n");
printf("I    8. Denormalize                   I₩n");
printf("I    9. Image convolution             I₩n");
printf("I    10. Quit                         I₩n");
printf("=====================================₩n");
printf("Enter number: ");
scanf("%d", &num);
return num;
```

do-while문이 반복될 때 처음으로 menu를 나타나게 해주며 num값을 입력받아 return 해주는 역할을 한다.

image_load() 파일 입력받고 이름 .ppm 지우기

```
images[i] = (CUBIC*)malloc(sizeof(CUBIC));
char fname[100] = { '₩0' };
printf("Enter image filename: ");
scanf("%s", fname);

FILE *infile;
infile = fopen(fname, "r");
if (infile == NULL)
{
printf("Could not open file₩n");
return;
}

int k = 0;
while (fname[k] != '₩0')
{
k++;
}
fname[k - 1] = '₩0'
fname[k - 2] = '₩0'
fname[k - 3] = '₩0'
fname[k - 4] = '₩0'
for (int t = 0; t < k - 1; t++)
images[i]->name[t] = fname[t];
```

image_load() 동적할당 및 파일에 있는 값 scan

```
char P;
int integ;
fscanf(infile, "%c%d %d %d %d", &P, &images[i]->C, &images[i]->W, &images[i]->H,
&integ);

images[i]->data = (float***)malloc(sizeof(float**) * (images[i]->H));
{
for (int a = 0; a < (images[i]->H); a++)
{
images[i]->data[a] = (float**)malloc(sizeof(float*) * (images[i]->W));
for (int b = 0; b < images[i]->W; b++)
{
images[i]->data[a][b] = (float*)malloc(sizeof(float) * (images[i]->C));
}
}
}

for (int a = 0; a < (images[i]->H); a++)
{
for (int b = 0; b < (images[i]->W); b++)
{
for (int c = 0; c < (images[i]->C); c++)
{
fscanf(infile, "%f", &(images[i]->data[a][b][c]));
}
}
}
images[i]->is_normalized = 0;
fclose(infile);
```

show_il()

```
for(int a = 0; a < i a++)
{
if (images[a]->is_normalized == 0)
{
int t = 0;
printf("%d. ", a);
while (images[a]->name[t] != '₩0')
{
printf("%c", images[a]->name[t]);
t++;
}
printf(" [H: %d, W: %d, C:%d]₩n", images[a]->H, images[a]->W, images[a]->C);
}
else
{
int t = 0;
printf("%d. ", a);
while (images[a]->name[t] != '₩0')
{
printf("%c", images[a]->name[t]);
t++;
}
printf(" [H: %d, W: %d, C:%d] - normalized₩n", images[a]->H, images[a]->W,
images[a]->C);
}
}
```

현재까지 있는 이미지들을 보여주는 역할을 한다.

show_fl()

```
for(int i = 0; i < j i++)
{
int t = 0;
printf("%d. ", i);
while (filters[i]->name[t] != '₩0')
{
printf("%c", filters[i]->name[t]);
t++;
}
printf(" [H: %d, W: %d, C:%d]₩n", filters[i]->H, filters[i]->W, filters[i]->C);
}
```

현재까지 있는 필터들을 보여주는 역할을 한다.

print()

```
printf("============ IMAGE LIST ==========₩n");
show_il(images, i);
printf("============ FILTER LIST ==========₩n");
show_fl(filters, j);
```

현재까지 있는 이미지와 필터의 목록을 출력한다.

image_save()

```
int num;
char ofname[100];
show_il(images, i);
printf("Enter the number of image to save: ");
scanf("%d", &num);

if (num < 0 || num >= i)
{
printf("ERROR print right number");
return;
}
printf("Enter filename for save image : ");
scanf("%s", ofname);

FILE *outfile;
outfile = fopen(ofname, "w");
fprintf(outfile, "P%d %d %d 255₩n", images[num]->C, images[num]->W, images[num]->H);
for (int a = 0; a < (images[num]->H); a++)
{
for (int b = 0; b < (images[num]->W); b++)
{
for (int c = 0; c < (images[num]->C); c++)
{
if (images[num]->data[a][b][c] < 0)
{
images[num]->data[a][b][c] = 0;
}
else if (images[num]->data[a][b][c] > 255)
{
images[num]->data[a][b][c] = 255;
}
fprintf(outfile, "%d ", (int)images[num]->data[a][b][c]);
}
}
}
fclose(outfile);
```

불러들인 이미지를 작업하거나 작업하지 않고 새로 저장하는 함수이다.

filter_load() 파일 입력받고 이름 .ppm 지우는 부분

```
filters[j] = (CUBIC*)malloc(sizeof(CUBIC));
char fname[100] = { '₩0' };
printf("Enter filter filename: ");
scanf("%s", fname);

FILE *infile;
infile = fopen(fname, "r");
if (infile == NULL)
{
printf("Could not open file₩n");
return;
}

int k = 0;
while (fname[k] != '₩0')
{
k++;
}
fname[k - 1] = '₩0'
fname[k - 2] = '₩0'
fname[k - 3] = '₩0'
fname[k - 4] = '₩0'
for (int t = 0; t < k - 1; t++)
filters[j]->name[t] = fname[t];
```

filter_load() 동적할당 및 pixel값 채우기

```
char P;
int integ;
fscanf(infile, "%c%d %d %d %d", &P, &filters[j]->C, &filters[j]->W, &filters[j]->H, &integ);

filters[j]->data = (float***)malloc(sizeof(float**) * (filters[j]->H));
{
for (int a = 0; a < (filters[j]->H); a++)
{
filters[j]->data[a] = (float**)malloc(sizeof(float*) * (filters[j]->W));
for (int b = 0; b < filters[j]->W; b++)
{
filters[j]->data[a][b] = (float*)malloc(sizeof(float) * (filters[j]->C));
}
}
}

for (int a = 0; a < (filters[j]->H); a++)
{
for (int b = 0; b < (filters[j]->W); b++)
{
for (int c = 0; c < (filters[j]->C); c++)
{
fscanf(infile, "%f", &(filters[j]->data[a][b][c]));
}
}
}
fclose(infile);
```

image_remove()

```
int num;
show_il(images, i);
printf("Enter the number of image to remove: ");
scanf("%d", &num);
if (num < 0 || num >= i)
{
printf("ERROR print right number");
return;
}

move_image_index(images, i, num);

for (int a = 0; a < (images[i - 1]->H); a++)
{
for (int b = 0; b < (images[i - 1]->W); b++)
{
free(images[i - 1]->data[a][b]);
}
free(images[i - 1]->data[a]);
}
free(images[i - 1]->data);
free(images[i - 1]);
```

사용자에게 받은 index값에 해당되는 이미지를 삭제해주는 역할을 한다.

filter_remove()

```
int num;
show_fl(filters, j);
printf("Enter the number of filter to remove: ");
scanf("%d", &num);
if (num < 0 || num >= j)
{
printf("ERROR print right number");
return;
}

move_filter_index(filters, j, num);

for (int a = 0; a < (filters[j - 1]->H); a++)
{
for (int b = 0; b < (filters[j - 1]->W); b++)
{
free(filters[j - 1]->data[a][b]);
}
free(filters[j - 1]->data[a]);
}
free(filters[j - 1]->data);
free(filters[j - 1]);
```

사용자에게 입력받은 index값에 해당되는 필터를 삭제하는 역할을 한다.

move_image_index()

```
for (int a = num a < i - 1; a++)
{
CUBIC *temp;
temp = images[a];
images[a] = images[a + 1];
images[a + 1] = temp;
}
```

image_remove를 불러왔을 때 이미지 인덱스를 순서대로 압축한다.

move_filter_index

```
for (int a = num a < j - 1; a++)
{
CUBIC *temp;
temp = filters[a];
filters[a] = filters[a + 1];
filters[a + 1] = temp;
}
```

filter_remove를 불러왔을 때 이미지 인덱스를 순서대로 압축한다.

normalize()

```
int num;
show_il(images, i);
printf("Enter the number of image to normalize: ");
scanf("%d", &num);
if (num < 0 || num >= i)
{
printf("ERROR print right number");
return;
}
if (images[num]->is_normalized == 1)
{
printf("ERROR already normalized₩n");
return;
}
images[num]->is_normalized = 1;
for (int a = 0; a < (images[num]->H); a++)
{
for (int b = 0; b < (images[num]->W); b++)
{
for (int c = 0; c < (images[num]->C); c++)
{
images[num]->data[a][b][c] = images[num]->data[a][b][c] / 128 - 1;
}
}
}
```

정규화하는 함수이다.

denormalize()

```
int num;
show_il(images, i);
printf("Enter the number of image to denormalize: ");
scanf("%d", &num);
if (num < 0 || num >= i)
{
printf("ERROR print right number");
return;
}
if (images[num]->is_normalized == 0)
{
printf("ERROR already denormalized₩n");
return;
}
images[num]->is_normalized = 0;
for (int a = 0; a < (images[num]->H); a++)
{
for (int b = 0; b < (images[num]->W); b++)
{
for (int c = 0; c < (images[num]->C); c++)
{
images[num]->data[a][b][c] = 128 * ((images[num]->data[a][b][c]) + 1);
}
}
}
```

역정규화 하는 함수이다.

image_convolution() 컨볼류젼 할 index를 입력받고 동적할당 하는 부분

```
int ic, fc;
int rh, rw;
float sum = 0;
show_il(images, i);
printf("Enter the number of image to convolve: ");
scanf("%d", &ic);
if (ic < 0 || ic >= i)
{
printf("ERROR print right number");
return;
}
show_fl(filters, j);
printf("Enter the number of filter to convolve: ");
scanf("%d", &fc);
if (fc < 0 || fc >= j)
{
printf("ERROR print right number");
return;
}

rh = images[ic]->H - filters[fc]->H + 1;
rw = images[ic]->W - filters[fc]->W + 1;

images[i] = (CUBIC*)malloc(sizeof(CUBIC));

images[i]->data = (float***)malloc(sizeof(float**) * rh);
{
for (int a = 0; a < rh; a++)
{
images[i]->data[a] = (float**)malloc(sizeof(float*) * rw);
for (int b = 0; b < rw; b++)
{
images[i]->data[a][b] = (float*)malloc(sizeof(float) * (images[ic]->C));
}
}
}
```

image_convolution()

```
for (int c = 0; c < 3; c++)
{
for (int arh = 0; arh < rh; arh++)
{
for (int arw = 0; arw < rw; arw++)
{
for (int h = 0; h < filters[fc]->H; h++)
{
for (int w = 0; w < filters[fc]->W; w++)
{
sum += (images[ic]->data[arh + h][arw + w][c]) * (filters[fc]->data[h][w][0]);
images[i]->data[arh][arw][c] = sum;
}
}
sum = 0;
}
}
}
if (images[ic]->is_normalized == 0)
{
images[i]->is_normalized = 0;
}
else
{
images[i]->is_normalized = 1;
}

images[i]->H = rh;
images[i]->W = rw;
images[i]->C = images[ic]->C;
```

값을 입력하고 is_normalized, H, W, C를 입력하는 부분이다.

image_convoltion()

```
char* new_name;
char* real_name;
char icname[100];
char* copy_name = strcpy(icname, images[ic]->name);
char bar[] = "_"
new_name = strcat(icname, bar);
real_name = strcat(new_name, filters[fc]->name);

int leng;
leng = strlen(images[ic]->name) + strlen(filters[fc]->name) + 1;

for (int t = 0; t < leng; t++)
images[i]->name[t] = real_name[t];
images[i]->name[leng] = NULL
```

컨볼루젼 연산 이후에 이름을 새롭게 입력하는 부분이다.

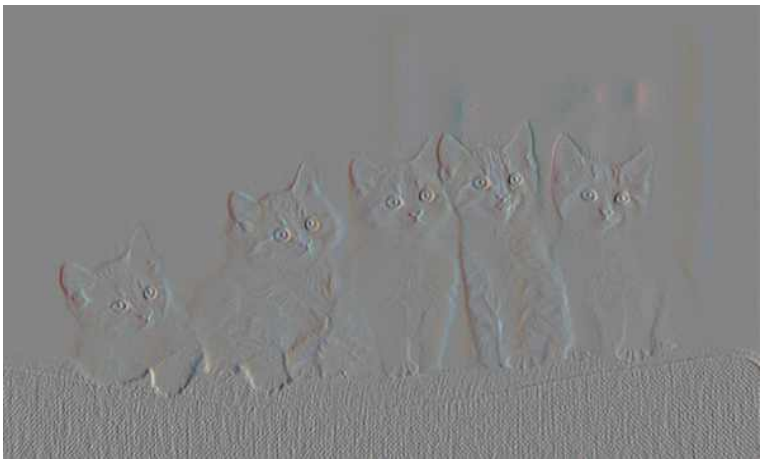
quit()

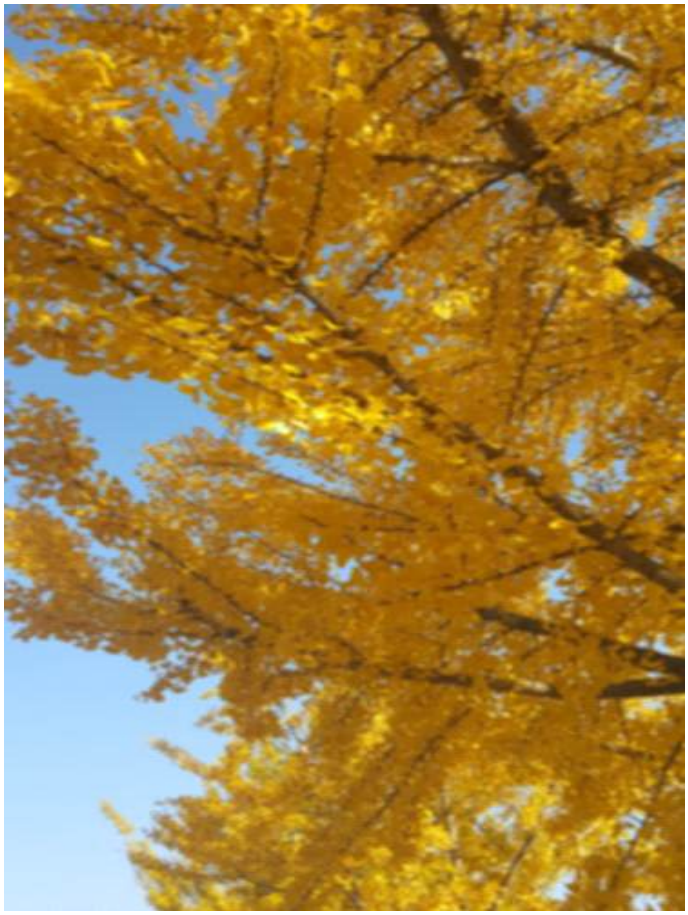
```
for (int q = 0; q < i q++)
{
for (int a = 0; a < (images[q]->H); a++)
{
for (int b = 0; b < (images[q]->W); b++)
{
free(images[q]->data[a][b]);
}
free(images[q]->data[a]);
}
free(images[q]->data);
free(images[q]);
}
free(images);
for (int p = 0; p < j p++)
{
for (int a = 0; a < (filters[p]->H); a++)
{
free(filters[p]->data[a]);
}
free(filters[p]->data);
free(filters[p]);
}
free(filters);
```

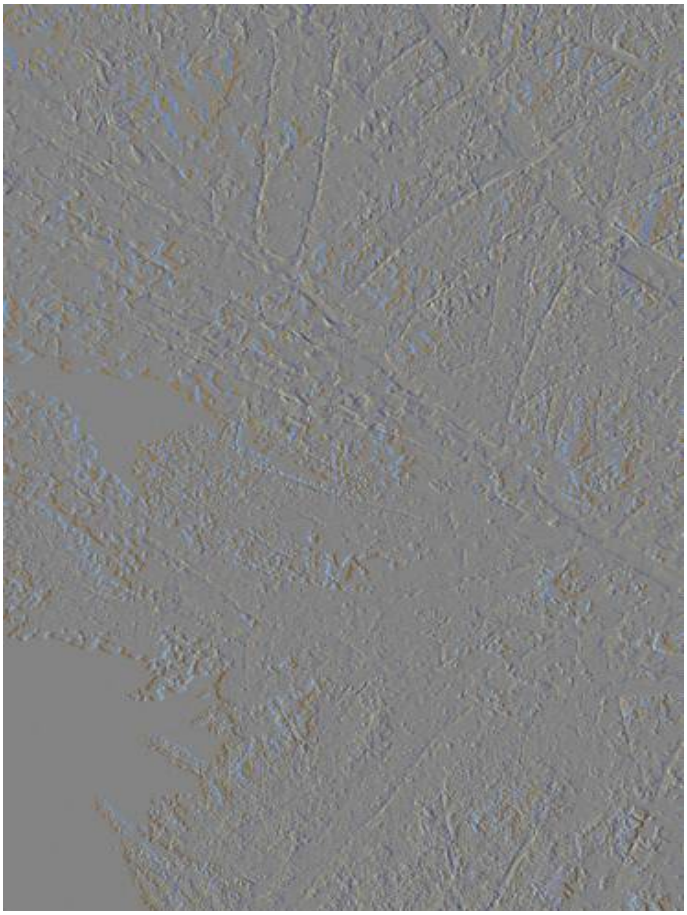
6. 프로그램 실행 방법

num 값으로 1을 입력하고 입력하고 싶은 image파일 명을 입력하여 image를 load 시킨다. 또한 num 값으로 4를 입력하고 입력하고 싶은 filter명을 입력하여 filter을 load 시킨다. 이미지를 삭제하고 싶을 때는 3번을 누르고 삭제하고 싶은 파일의 index를 입력하면 되고, filter을 삭제하고 싶을 때는 5번을 누르고 삭제하고 싶은 파일의 index를 입력하면 된다. 6번을 누르면 지금까지 있는 이미지와 필터의 목록이 출력되며 7번을 누르면 정규화가 되고, 8번을 누르면 역정규화가 되며 9번을 누르면 입력한 이미지와 필터가 컨볼류젼이 된다. 이렇게 연산을 거친 파일을 2번을 눌러서 save시키면 파일이 저장되며 10번을 누르면 프로그램이 종료된다.

7. 프로그램 실행 예제







8. 토론 및 결론

이번 assignment를 통하여 동적 할당과 다차원 배열이라는 어려운 개념을 제대로 이해하는데 많은 도움이 된 것 같다. 또한 앞에서 배운 파일 입출력 부분도 다시 한 번 다뤄보며 잊어버리지 않게 되었고, 반복문, 조건문, 사용자 함수 생성 등 앞에서 배운 여러 가지 개념들도 적용시켜보며 지금까지 배운 개념들을 다잡는 계기가 된 것 같다. 또한 구조체도 새롭게 배우면서 이를 적용시켜 프로그램을 만들어 가며 이미지에 필터를 입히고 이를 확인하였을 때 희열을 느낄 수 있었다. 이번 프로그램을 통하여 평소 아무 생각도 없이 사용하던 필터의 원리를 이해할 수 있게 된 계기가 된 것 같다.