영상처리 실습 7

# 출력 결과

**실습 1.**

<lena.pgm / bg.pgm >

사람, 여자, 의류, 실외이(가) 표시된 사진

자동 생성된 설명흐림이(가) 표시된 사진

자동 생성된 설명

< lena.pgm + bg.pgm / lena.pgm – bg.pgm >

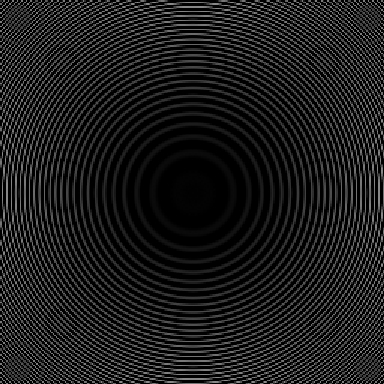
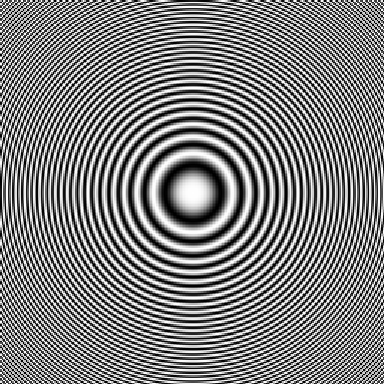
의류, 사람, 여자, 머리장식이(가) 표시된 사진

자동 생성된 설명사람, 여자, 모자, 머리카락이(가) 표시된 사진

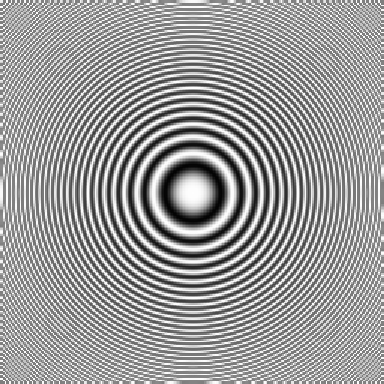
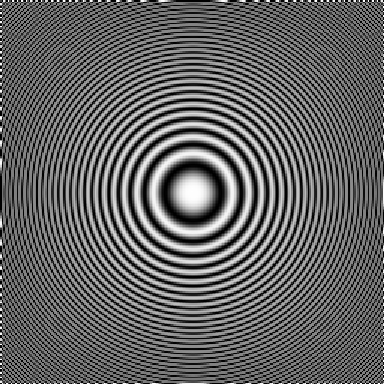
자동 생성된 설명

**실습 2.**

< czp.pgm / czp.pgm – czp\_lpf.pgm >



< czp.pgm – czp\_hpf.pgm / czp\_lpf.pgm + czp\_hpf.pgm >



**실습 3.**

< lena.bmp / lena\_mid.pgm >

사람, 여자, 의류, 하얀색이(가) 표시된 사진

자동 생성된 설명사람, 여자, 하얀색이(가) 표시된 사진

자동 생성된 설명

* lena\_mid.pgm 은 .bmp 파일을 순서 그대로 읽어 들인 후, .pgm 파일에 다시 쓴 결과.
* 위 / 아래로 반전되므로, .pgm 파일로 변환할 때, 반대로 써야 함.

<lena\_bmp\_to\_pgm.pgm >



# 코드 내용

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File: get\_bmphead.c \*

\* \*

\* Desc: This program reads header part of bmp format image \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include "ip.h"

typedef unsigned short int USHORT;

typedef unsigned long int ULONG;

struct BITMAPHEADER {

USHORT bmpType; // BMP 파일의 종류를 나타내는 식별자

ULONG bmpSize; // BMP 파일의 크기(바이트 단위)

USHORT bmpReserved1; // 예약

USHORT bmpReserved2; // 예약

ULONG bmpOffset; // 이미지 정보가 시작되는 위치(Offset)

ULONG bmpHeaderSize; // Header의 크기(바이트 단위)

ULONG bmpWidth; // 가로 크기(픽셀 단위)

ULONG bmpHeight; // 세로 크기(픽셀 단위)

USHORT bmpPlanes; // Color Plane의 수(1로 셋팅)

USHORT bmpBitCount; // 한 픽셀에 할당된 비트수

ULONG bmpCompression; // 압축 기법 사용여부

ULONG bmpBitmapSize; // 비트맵 이미지의 크기

ULONG bmpXPelsPerMeter; // 수평 해상도

ULONG bmpYPelsPerMeter; // 수직 해상도

ULONG bmpColors; // 사용된 색상수

ULONG bmpClrImportant;

};

struct BITMAPHEADER bmp\_header; // Bitmap 파일의 Header 구조체

void ReadBitmapHeader(char \*filename) {

int read\_fd; // 파일을 읽고 쓰는데 사용할 Descriptor

read\_fd = open(filename, O\_RDONLY); // bmp파일을 open한다.

if (read\_fd == -1) {

printf("Can't open file %s\n", filename);

exit(1);

}

read(read\_fd, &bmp\_header.bmpType, sizeof(bmp\_header.bmpType));

read(read\_fd, &bmp\_header.bmpSize, sizeof(bmp\_header.bmpSize));

read(read\_fd, &bmp\_header.bmpReserved1, sizeof(bmp\_header.bmpReserved1));

read(read\_fd, &bmp\_header.bmpReserved2, sizeof(bmp\_header.bmpReserved2));

read(read\_fd, &bmp\_header.bmpOffset, sizeof(bmp\_header.bmpOffset));

read(read\_fd, &bmp\_header.bmpHeaderSize, sizeof(bmp\_header.bmpHeaderSize));

read(read\_fd, &bmp\_header.bmpWidth, sizeof(bmp\_header.bmpWidth));

read(read\_fd, &bmp\_header.bmpHeight, sizeof(bmp\_header.bmpHeight));

read(read\_fd, &bmp\_header.bmpPlanes, sizeof(bmp\_header.bmpPlanes));

read(read\_fd, &bmp\_header.bmpBitCount, sizeof(bmp\_header.bmpBitCount));

read(read\_fd, &bmp\_header.bmpCompression, sizeof(bmp\_header.bmpCompression));

read(read\_fd, &bmp\_header.bmpBitmapSize, sizeof(bmp\_header.bmpBitmapSize));

read(read\_fd, &bmp\_header.bmpXPelsPerMeter, sizeof(bmp\_header.bmpXPelsPerMeter));

read(read\_fd, &bmp\_header.bmpYPelsPerMeter, sizeof(bmp\_header.bmpYPelsPerMeter));

read(read\_fd, &bmp\_header.bmpColors, sizeof(bmp\_header.bmpColors));

read(read\_fd, &bmp\_header.bmpClrImportant, sizeof(bmp\_header.bmpClrImportant));

close(read\_fd);

}

image\_ptr read\_bmp(char\* filename, int rows, int cols, int type, ULONG startOffset)

{

int i; /\* index variable \*/

int row\_size; /\* size of image row in bytes \*/

int maxval; /\* maximum value of pixel \*/

FILE\* fp; /\* input file pointer \*/

int firstchar, secchar; /\* first 2 characters in the input file \*/

image\_ptr ptr; /\* pointer to image buffer \*/

unsigned long offset; /\* offset into image buffer \*/

unsigned long total\_size; /\* size of image in bytes \*/

unsigned long total\_bytes; /\* number of total bytes written to file \*/

float scale; /\* number of bytes per pixel \*/

/\* open input file \*/

if ((fp = fopen(filename, "rb")) == NULL) // 예외 처리. 문제가 생길 시 프로그램이 에러를 출력하고 종료.

{

printf("Unable to open %s for reading\n", filename);

exit(1);

}

fseek(fp, startOffset, SEEK\_SET);

scale = 1.0;

row\_size = cols \* scale;

total\_size = (unsigned long)rows \* row\_size; // 전체 크기 계산.

ptr = (image\_ptr)IP\_MALLOC(total\_size); // 전체 크기만큼 메모리 할당.

if (ptr == NULL) // 예외 처리. ptr == NULL 이면, 메모리 할당이 정상적으로 이루어지지 않았음을 의미하기 때문에.

{

printf("Unable to malloc %lu bytes\n", total\_size);

exit(1);

}

total\_bytes = 0;

offset = 0;

for (i = 0; i < rows; i++) // 마지막 행까지 반복.

{

total\_bytes += fread(ptr + offset, 1, row\_size, fp); // ptr 배열에 1 byte (0 ~ 255) 크기의 값을 파일로부터 한 행씩 입력받음.

offset += row\_size; // 다음 행의 첫 offset 을 지정함.

}

if (total\_size != total\_bytes) // total\_size 와 total\_bytes 가 같은 지 확인함으로써, 파일로부터 데이터를 불러오는 데에 문제가 없었는지 확인.

{

printf("Failed miserably trying to read %ld bytes\nRead %ld bytes\n",

total\_size, total\_bytes);

exit(1);

}

fclose(fp);

return ptr; // ptr 의 첫 offset 주소 반환.

}

void convert\_bmp\_to\_pnm(image\_ptr ptr, char\* filename, int rows,

int cols, int magic\_number)

{

FILE\* fp; /\* file pointer for output file \*/

long offset; /\* current offset into image buffer \*/

long total\_bytes; /\* number of bytes written to output file \*/

long total\_size; /\* size of image buffer \*/

int row\_size; /\* size of row in bytes \*/

int i; /\* index variable \*/

float scale; /\* number of bytes per image pixel \*/

switch (magic\_number)

{

case 4: /\* PBM \*/

scale = 0.125;

break;

case 5: /\* PGM \*/

scale = 1.0;

break;

case 6: /\* PPM \*/

scale = 3.0;

break;

default: /\* Error \*/

printf("write\_pnm: This is not a Portable bitmap RAWBITS file\n");

exit(1);

break;

}

/\* open new output file \*/

if ((fp = fopen(filename, "wb")) == NULL)

{

printf("Unable to open %s for output\n", filename);

exit(1);

}

/\* print out the portable bitmap header \*/ // 헤더 작성

fprintf(fp, "P%d\n%d %d\n", magic\_number, cols, rows);

if (magic\_number != 4)

fprintf(fp, "255\n");

row\_size = cols \* scale;

total\_size = (long)row\_size \* rows; // 전체 크기 계산.

offset = total\_size - row\_size; // 원본 write\_pnm() 함수에서 수정된 부분.

total\_bytes = 0;

for (i = 0; i < rows; i++)

{

total\_bytes += fwrite(ptr + offset, 1, row\_size, fp); // ptr 배열로부터 1 byte (0 ~ 255) 크기의 값을 파일에 한 행씩 입력함.

offset -= row\_size; // 원본 write\_pnm() 함수에서 수정된 부분. 다음 행의 첫 offset 을 지정함.

}

if (total\_bytes != total\_size) // total\_size 와 total\_bytes 가 같은 지 확인함으로써, 파일에 데이터를 쓰는 데에 문제가 없었는지 확인.

printf("Tried to write %ld bytes...Only wrote %ld\n",

total\_size, total\_bytes);

fclose(fp);

}

int main(int argc, char\* argv[])

{

char filein[100]; /\* name of input file \*/

char fileout[100]; /\* name of output file \*/

int rows, cols; /\* image rows and columns \*/

unsigned long i; /\* counting index \*/

unsigned long bytes\_per\_pixel; /\* number of bytes per image pixel \*/

image\_ptr buffer; /\* pointer to image buffer \*/

// image\_ptr obuffer; /\* pointer to image buffer \*/

unsigned long number\_of\_pixels; /\* total number of pixels in image \*/

int type; /\* what type of image data \*/

/\* set input filename and output file name \*/

if (argc == 3)

{

strcpy(filein, argv[1]);

strcpy(fileout, argv[2]);

}

else

{

printf("Input name of input file\n");

gets(filein);

printf("\nInput name of output file\n");

gets(fileout);

printf("\n");

}

ReadBitmapHeader(filein);

cols = bmp\_header.bmpWidth;

rows = bmp\_header.bmpHeight;

type = PGM;

buffer = read\_bmp(filein, cols, rows, type, bmp\_header.bmpOffset);

/\* determine bytes\_per\_pixel, 3 for color, 1 for gray-scale \*/

if (type == PPM)

bytes\_per\_pixel = 3;

else

bytes\_per\_pixel = 1;

number\_of\_pixels = bytes\_per\_pixel \* rows \* cols;

convert\_bmp\_to\_pnm(buffer, fileout, rows, cols, type);

IP\_FREE(buffer);

return 0;

}