CSED101. Programming & Problem solving
Spring 2022

Programming Assignment #4
(70 points)

김건우 (geonu.kim@postech.ac.kr)

■ **제출 마감일: 2022.05.17 23:59**

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

■ **제출물**

- C 소스 코드 (assn4.c)
  - ➢ 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn4.docx, assn4.hwp 또는 assn4.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]

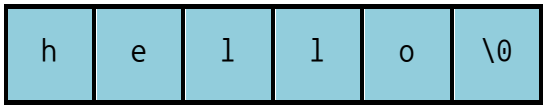
1. 문자열(string)

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

2. 선언

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

  str



3. 입출력 예시

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

```c
char filename[128];  // 최대 127 자 까지 저장 가능
FILE* fp;

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

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

<실행 예시> (아래의 빨간색 밑줄은 사용자 입력에 해당)

```
Enter the filename: cat.ppm
cat.ppm
```

■ Problem: 이미지 크기 변형하기

(개요)

쌍선형 보간을 이용해 이미지를 확대하거나 축소하는 프로그램을 작성하세요.


(목표)

- 포인터의 선언과 사용을 익힙니다.
- 다차원 배열의 동적 할당과 해제 및 사용을 익힙니다.
- 함수에서 포인터를 매개변수로 사용하는 방법을 익힙니다.
- 텍스트 파일 입출력을 익힙니다.

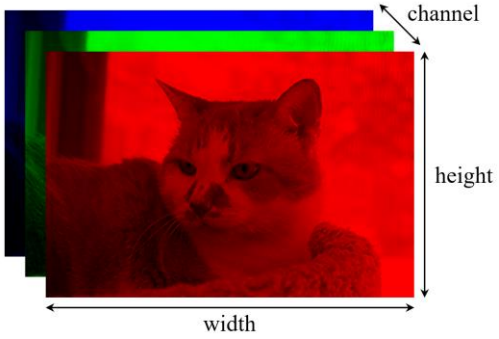
(주의사항)

1. 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. main() 함수에 모든 기능을 구현한 경우 감점 처리합니다.
2. 프로그램 구현 시, main() 함수를 호출을 직접 하지 않습니다. 즉, 소스 코드 내에 main(); 이라고 호출하지 않습니다.
3. 전역 변수, goto 문, 구조체는 사용할 수 없습니다.
4. 연결리스트(linked list)와 같은 다른 방법을 사용할 경우, 감점 처리합니다.
5. 프로그램 종료 시 모든 변수가 반드시 올바르게 할당 해제되어야 합니다.
6. 사용자 입력에서 숫자를 입력 받는 부분에는 숫자만 입력하는 것으로 가정합니다. 즉, 숫자 입력 받는 부분에는 문자 등의 입력에 대해서는 고려할 필요가 없습니다.
7. 명시된 에러 처리 외에는 고려하지 않아도 됩니다.
8. 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) 3차원으로 구성된 배열로 표현됩니다. 일반적으로 이미지는 빛의 3원색인 빨강, 초록, 파랑으로 구성되어 총 3개의 채널을 가지고 있습니다. 많은 이미지들이 효율성을 위해 각 데이터를 1 Byte로 저장하기 때문에, 한 픽셀의 값은 0부터 255 사이의 정수 3개로 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.

image.ppm

```
P3 4 3 255
1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36
```

PPM 이미지에서 가장 먼저 나오는 ‘P3’은 하나의 픽셀이 3개의 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.

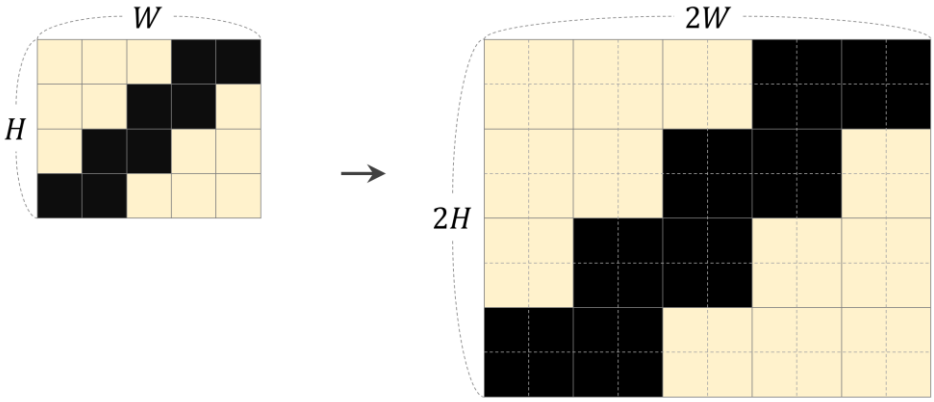
나머지는 각각의 픽셀 데이터가 채널, 너비, 높이 순으로 나열되어 있습니다. 예를 들어 위의 예제에서 첫 4개의 값을 제외한 가장 앞의 ‘1 2 3’은 왼쪽 위 꼭짓점에 위치한 픽셀의 RGB 값을 뜻하고, 이어지는 ‘4 5 6’은 그 오른쪽 픽셀의 RGB 값을 뜻합니다.

ppm 형식 파일은 이미지 뷰어인 ‘꿀뷰’를 사용해서 이미지 확인이 가능하며, 제공되는 ppm 파일은 ASCII code로 작성되었기 때문에 텍스트 편집기로 픽셀 값을 확인할 수 있습니다.

- 꿀뷰 다운로드: https://kr.bandisoft.com/honeyview/)

(쌍선형 보간)

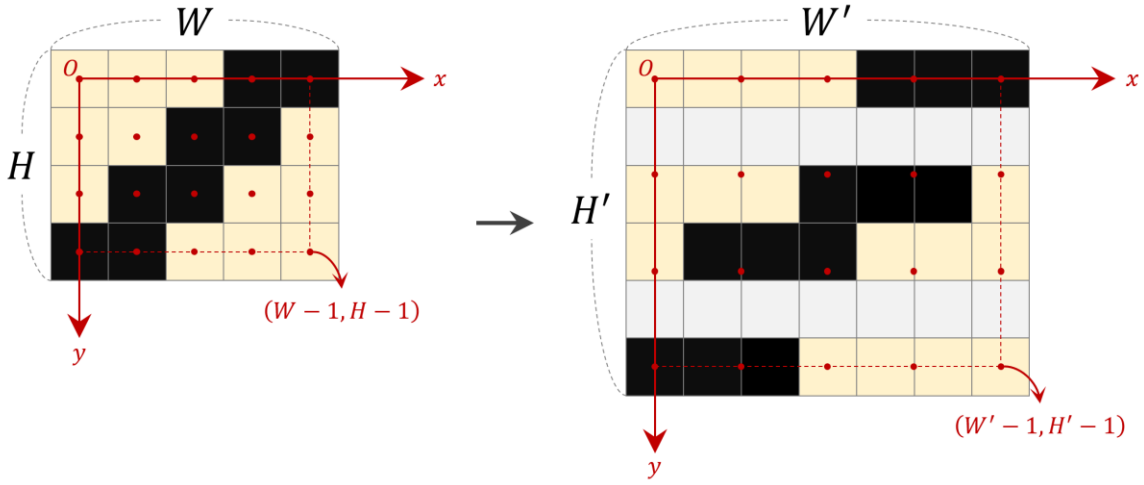
이미지를 확대하는 가장 간단한 방법은 확대하고자 하는 배율만큼 픽셀을 복제하는 것입니다. 그림판에서 이미지를 열고 확대하면 이와 같은 방법으로 화면에 표시됩니다.



만약 정수배가 아닌 크기로 이미지를 확대하려면 어떻게 해야 할까요? 이미지 크기를 조절하는 문제를 수학적으로 일반화하기 위해 다음과 같은 과정을 거칩니다.

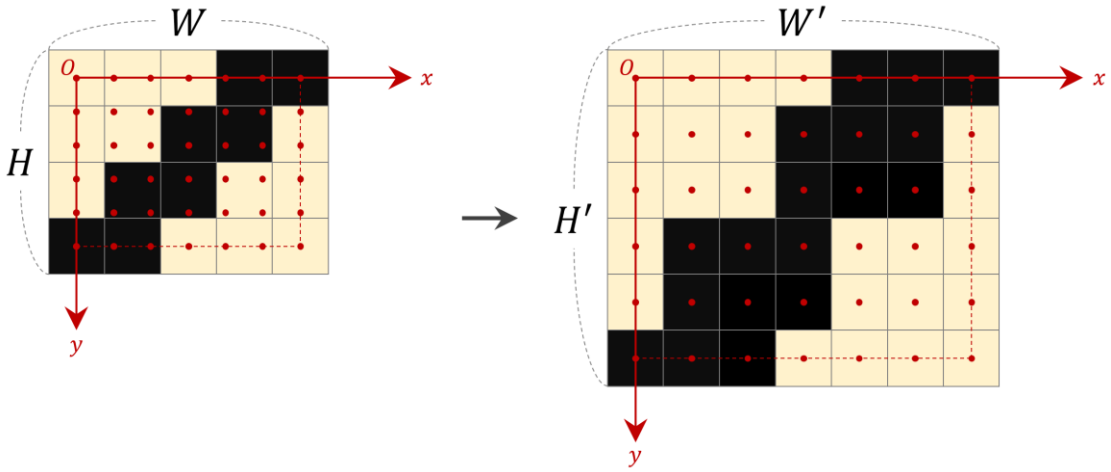
1. 이미지의 너비(width)와 높이(height)를 각각 W, H로 정의합니다.
2. 이미지의 왼쪽 위 꼭짓점에 있는 픽셀의 좌표를 $(0, 0)$로 정의합니다.
3. 오른쪽 아래 꼭짓점에 있는 픽셀의 좌표를 $(W-1, H-1)$으로 정의합니다.

이 때, 이미지는 이웃한 점 사이의 거리가 1인 점들의 집합으로 볼 수 있습니다. $W \times H$ 해상도의 이미지를 $W' \times H'$ 해상도로 변환하는 것은 다음 그림과 같이 각각의 픽셀을 $(x', y') = \left(\frac{W'-1}{W-1}x, \frac{H'-1}{H-1}y\right)$ 의 좌표로 이동시키는 것으로 생각할 수 있습니다.



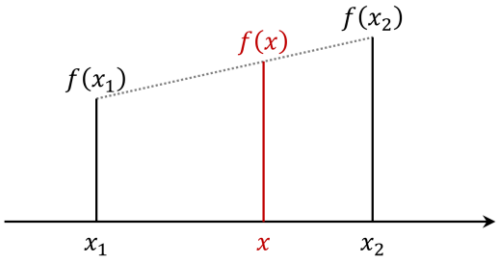
위의 그림에서 볼 수 있듯이, 이동된 픽셀 좌표는 정수값이 아니기 때문에 출력 이미지의 픽셀에 일대일 대응되지 않습니다. 특히 이미지를 확대할 경우 원본 이미지의 픽셀보다 변환 후 이미지의 픽셀 수가 더 많기 때문에 모든 픽셀을 채울 수 없습니다.

이런 문제를 해결하기 위해, 역으로 출력 이미지의 각 (x', y') 좌표에 대한 원본 이미지 좌표 $(x, y) = \left(\frac{W-1}{W'-1}x', \frac{H-1}{H'-1}y'\right)$를 찾아 해당 픽셀의 값을 사용하며 이를 역방향 사상(backward mapping)이라고 합니다.



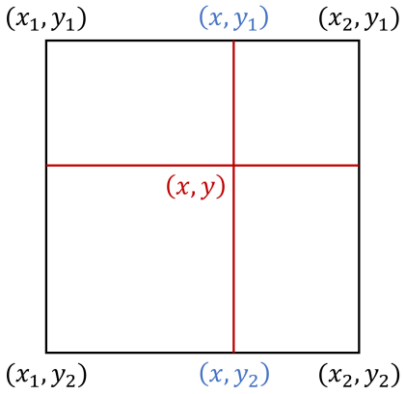
역방향 사상 된 (x,y) 좌표 또한 정수값이 아니기 때문에, 값을 반올림해서 가장 가까운 픽셀의 값을 사용하며 이러한 방법을 최단 입점 보간(nearest neighbor interpolation)이라고 합니다. 최단 입점 보간으로 이미지를 확대하면 픽셀 한 개가 이루는 사각 영역이 너무 커져서 계단 현상이 발생하며, 특히 정수가 아닌 배율로 확대할 때 사각 영역의 크기가 일정하지 않아 이미지가 왜곡됩니다. 이러한 문제를 개선하는 여러가지 방법이 고안되었고, 이번 과제에서는 그 중 가장 간단한 방법인 쌍선형 보간(bilinear interpolation)을 사용하려고 합니다.

선형 보간은 직선상의 두 점 x_1,x_2와 함숫값 $f(x_1),f(x_2)$가 존재할 때, 두 점 사이 임의의 점 x의 함숫값 $f(x)$를 다음과 같이 거리에 비례한 값으로 정의합니다.



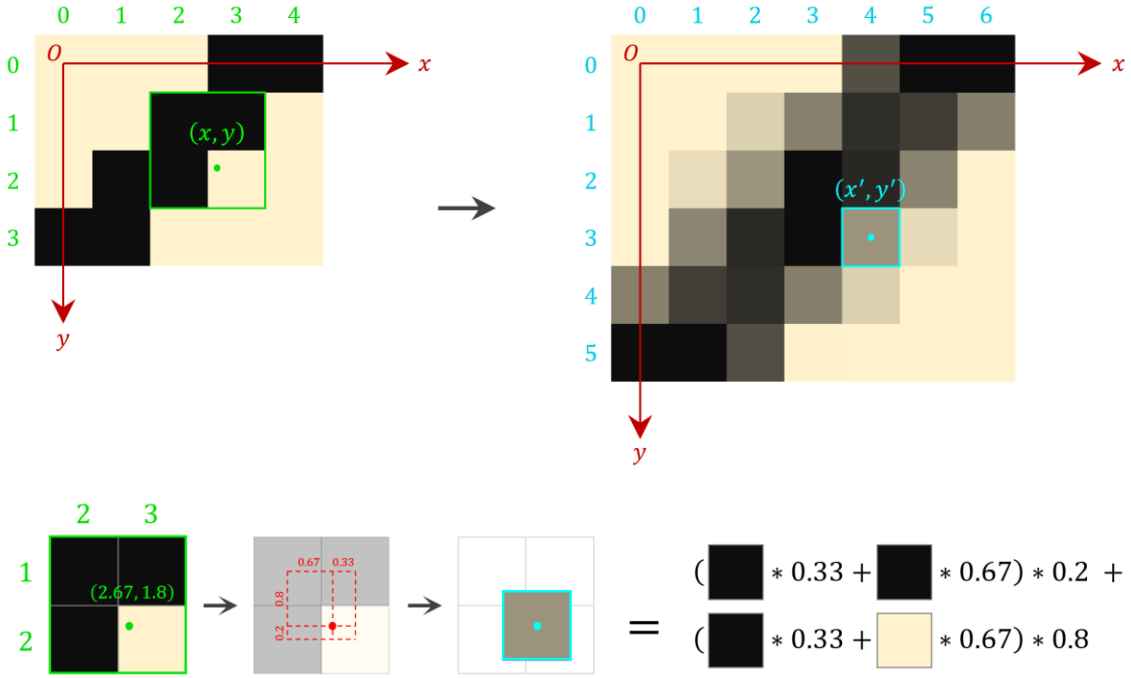
$$f(x) = f(x_1) + \big(f(x_2) - f(x_1)\big)\frac{x - x_1}{x_2 - x_1} \quad \Leftrightarrow \quad \frac{f(x) - f(x_1)}{x - x_1} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

쌍선형 보간은 선형 보간을 2차원으로 확장한 것으로, 총 4개의 이웃한 점에 대해 다음과 같이 x,y축 각각 선형 보간을 적용해 함숫값을 구합니다.



$$f(x,y_1) = \frac{x_2 - x}{x_2 - x_1} f(x_1,y_1) + \frac{x - x_1}{x_2 - x_1} f(x_2,y_1)$$
$$f(x,y_2) = \frac{x_2 - x}{x_2 - x_1} f(x_1,y_2) + \frac{x - x_1}{x_2 - x_1} f(x_2,y_2)$$
$$f(x,y) = \frac{y_2 - y}{y_2 - y_1} f(x,y_1) + \frac{y - y_1}{y_2 - y_1} f(x,y_2)$$

예시와 함께 쌍선형 보간을 이미지 크기 조절에 적용해보겠습니다. 다음과 같이 $4 * 5$ 크기의 원본 이미지 src를 $6 * 7$ 크기의 출력 이미지 dst로 확대한다고 합시다. $dst[4,3]$를 구할 때, 이와 대응되는 원본 픽셀 좌표는 역방향 사상을 적용하면 $src\left[4 * \frac{4}{6}, 3 * \frac{3}{5}\right] = src[2.67, 1.8]$ 임을 알 수 있습니다. $x = 2.67, y = 1.8$ 은 정수값이 아니므로, 이웃한 픽셀 값에 쌍선형 보간을 적용해 출력 픽셀 값을 계산해야 합니다. $x_1 \le x \le x_2$ 를 만족하는 가장 가까운 x_1는 2.67을 내림한 2이고, x_2는 x_1에 1을 더한 3이 됩니다. 같은 방법으로 $y_1 = \lfloor y \rfloor = 1$, $y_2 = y_1 + 1 = 2$ 임을 알 수 있습니다. 쌍선형 보간 수식에 이를 적용하면, $src[2,1]$, $src[2,2]$, $src[3,1]$, $src[3,2]$ 값으로부터 $src[2.67, 1.8]$ 의 값을 구할 수 있습니다.



쌍선형 보간은 이미지를 확대할 때뿐만 아니라 축소할 때도 같은 방법으로 적용할 수 있습니다. 이번 과제에서는 자유로운 크기로 이미지 크기를 조절하는 프로그램을 만들어야 합니다.

*** 참고사항**

과제를 구현할 때 위와 같이 4개의 인접한 픽셀을 항상 참조하면 이미지 테두리 부분의 값을 구할 때 원본 이미지 배열의 범위 밖을 참조하게 됩니다. 예를 들어 위의 예시에서 $dst[4,5]$를 계산할 때, 대응되는 원본 좌표는 $src[2.67, 3]$ 이므로 $src[2,3]$, $src[2,4]$, $src[3,3]$, $src[3,4]$ 를 참조하고 싶지만 $src[2,4]$, $src[3,4]$ 는 배열의 범위를 벗어나므로 메모리 오류가 발생하게 됩니다. 쌍선형 보간의 수식을 잘 생각해보면 $src[2.67, 3] = 0.33 * src[2,3] + 0.67 * src[3,3]$ 으로 2개의

픽셀 값만 사용하기 때문에, 배열의 범위를 벗어난 위치를 참조할 필요가 없습니다. 이 점을 고려하여, 구현 시 배열의 범위를 벗어난 픽셀을 참조해서 메모리 오류가 발생하는 일이 발생하지 않도록 주의합니다.

II. 구현 요구사항

이번 과제에서 구현해야 할 것은 다음과 같습니다.

- 다차원 배열을 동적으로 할당하고 해제합니다.
- PPM 이미지를 불러오거나 저장합니다.
- 원하는 배율을 입력받아 이미지에 쌍선형 보간을 적용합니다.

(메뉴 화면)

프로그램을 시작했을 때 다음과 같이 선택 메뉴를 출력하고, 사용자 입력을 받아 해당 기능을 수행합니다. 하나의 기능을 완료하거나 에러 발생 시 다시 메뉴가 출력되도록 합니다.

```
============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number:
```

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

*** 에러 처리** (11쪽 출력 예시 참고)

- 존재하지 않는 메뉴를 선택했을 경우 'Invalid menu selected'를 출력합니다.

(이미지 불러오기)

'1'을 입력할 경우 아래와 같이 사용자로부터 이미지 파일 이름을 입력 받고, 이미지가 유효한 경우 이미지 파일로부터 데이터를 읽어 동적 할당 받은 배열에 저장합니다. 먼저 불러온 이미지가 있는 경우에는 기존 이미지를 삭제(동적 할당 해제)하고 새로운 이미지를 저장하도록 합니다.

```
============================
Enter number: 1
Enter input filename: raccoon.ppm
Read raccoon.ppm (533x800)

============================
|      IMAGE RESIZER       |
```

(예시의 노란색 글씨는 사용자 입력을 뜻합니다.)

파일 이름은 공백을 포함하지 않으며, 100자를 넘지 않는다고 가정합니다.

파일의 위치는 프로그램이 실행되고 있는 폴더로, Visual Studio와 Dev-C++ 에디터에서 프로그램 실행 시 별다른 설정이 없을 경우 소스코드(assn4.c)와 같은 폴더에 위치합니다.

*** 에러 처리**

- 파일이 존재하지 않을 경우 “File not exists”를 출력합니다.
- 이미지 형식이 잘못되었을 경우, 즉 픽셀 값이 주어진 크기만큼 존재하지 않거나 0~255 사이 범위를 벗어나는 경우 “Image corrupted”를 출력합니다.

이미지는 동적 할당으로 필요한 크기의 배열을 생성해 저장합니다. 아래는 주어진 크기의 2차원 배열을 동적으로 할당하는 예시입니다.

```
// red 채널 이미지 저장을 위한 동적 할당(이미지의 width, height 가 주어졌다고 가정)
int **red;
int i;

red = (int **)malloc(sizeof(int **) * height);
for (i = 0; i < height; i++)
    *(image + i) = (int *)malloc(sizeof(int) * width);
```

(이미지 저장하기)

‘2’를 입력할 경우 아래와 같이 사용자로부터 저장할 이미지 파일 이름을 입력 받고, 배열에 저장된 이미지를 파일로 출력합니다.

```
==============================
Enter number: 2
Enter output filename: raccoon.ppm
Saved raccoon.ppm (533x800)

==============================
|       IMAGE RESIZER        |
```

*** 에러 처리**

- 저장할 이미지가 없는 경우, 즉 ‘이미지 불러오기’를 한 적이 없는 경우 “Image is not loaded”를 출력합니다.

(이미지 크기 조절하기)

‘3’을 입력할 경우 아래와 같이 사용자로부터 출력 이미지 크기를 입력 받고, 앞서 설명한 쌍선형 보간을 사용해 이미지 크기를 변형합니다. 출력 이미지의 크기가 원본 이미지 크기와 다르기 때문에 새로운 배열을 알맞게 동적 할당해야 하며, 이후 필요 없어진 원본 이미지 배열은 올바르게 할당 해제해야 합니다.

```
==============================
Enter number: 3
Enter output size: 260 400
Resized image (533x800 -> 260x400)

==============================
|       IMAGE RESIZER        |
```

*** 에러 처리**

- 불러온 이미지가 없을 경우 “Image is not loaded”를 출력합니다.

- 입력받은 이미지 크기가 양수가 아닐 경우 “Invalid size”를 출력합니다.

(프로그램 종료)

‘0’을 입력할 경우 프로그램을 종료합니다. 종료 시 동적 할당된 모든 메모리를 올바르게 할당 해제해야 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, 매개변수의 개수와 자료형, 반환 자료형 등은 자유롭게 변경이 가능합니다. 이 때 무엇을 어떻게 변경해서 구현했는지 보고서에 기록하도록 합니다.

아래 명시된 함수는 이미지를 3차원 배열 포인터(int ***)로 표현하였으나, 편의를 위해 3개의 2차원 배열 포인터(int **)로 각 색상 정보를 따로 저장해도 무방합니다.

```
int ***create_image(int width, int height);
```

- 이미지 크기를 전달받아 알맞은 크기의 배열을 동적 할당하는 함수입니다.

```
void delete_image(int ***image);
```

- 동적 할당된 배열을 할당 해제하는 함수입니다.

```
int load_image(const char *filename, int ****image, int *width, int *height);
```

- 이미지를 읽고 이미지의 정보를 전달받은 포인터에 알맞게 전달합니다.
- 알맞은 크기의 배열을 동적 할당해야 합니다.

```
int save_image(const char *filename, int ***image, int width, int height);
```

- 전달받은 이미지를 파일로 저장하는 함수입니다.

```
int resize_image(int ***src, int src_width, int src_height,
                 int ***dst, int dst_width, int dst_height);
```

- src로 전달받은 이미지를 쌍선형 보간 적용 후 dst에 출력하는 함수입니다.
- src, dst의 동적 할당과 해제는 함수 내에서 수행하지 않습니다.
- 쌍선형 보간은 실수 연산으로 수행하는 반면, ppm 이미지는 0~255 범위의 정수로 저장됩니다. 따라서 연산 결과가 0보다 작을 경우 0으로, 255보다 클 경우 255로 대체하며, 소숫점 아래는 반올림합니다.

(출력 예시)

다음은 프로그램의 실행 화면과 출력된 이미지 예시입니다.

```
============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 5
Invalid menu selected

============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 2
Image is not loaded

============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 1
Enter input filename: raccoon.ppm
Read raccoon.ppm (533x800)

============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 3
Enter output size: 1000 1000
Resized image (533x800 -> 1000x1000)

============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 2
Enter output filename: raccoon_1000x1000.ppm
Saved raccoon_1000x1000.ppm (1000x1000)

============================
|      IMAGE RESIZER       |
|  1. Load      2. Save    |
|  3. Resize    0. Quit    |
============================
Enter number: 0
Quit program

계속하려면 아무 키나 누르십시오 . . .
```

raccoon.ppm

raccoon_1000x1000.ppm