科目名: プログラミング実習III

課題: B4

番号: 27020856

氏名: 趙 聖化

作成: 2023年10月17日

----------------------------------------

[課題 4.1]

(1)matrix\_print

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix1test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out

A

0.00000 1.00000 2.00000

1.00000 2.00000 3.00000

2.00000 3.00000 4.00000

chosunghwa@joseonghwaui-MacBookAir 4-1 % diff -w matrix1print-out.txt matrixprint-exp.txt

chosunghwa@joseonghwaui-MacBookAir 4-1 %

(2)matrix\_scan

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix2test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out

n = 3

[0][0] 1

[0][1] 2

[0][2] 3

[1][0] 4

[1][1] 5

[1][2] 6

[2][0] 7

[2][1] 8

[2][2] 9

A

1.00000 2.00000 3.00000

4.00000 5.00000 6.00000

7.00000 8.00000 9.00000

(3)matrix\_sum

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix3test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out < in.txt > out.txt

n = A

[0][0] [0][1] [0][2] [0][3] [0][4] [1][0] [1][1] [1][2] [1][3] [1][4] [2][0] [2][1] [2][2] [2][3] [2][4] [3][0] [3][1] [3][2] [3][3] [3][4] [4][0] [4][1] [4][2] [4][3] [4][4] B

[0][0] [0][1] [0][2] [0][3] [0][4] [1][0] [1][1] [1][2] [1][3] [1][4] [2][0] [2][1] [2][2] [2][3] [2][4] [3][0] [3][1] [3][2] [3][3] [3][4] [4][0] [4][1] [4][2] [4][3] [4][4] %

chosunghwa@joseonghwaui-MacBookAir 4-1 % diff -w out.txt exp.txt

chosunghwa@joseonghwaui-MacBookAir 4-1 %

(4)matrix\_trans2

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix4test.c

^[[A%

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out < trans2-in.txt > trans2-out.txt

n = A

[0][0] [0][1] [0][2] [0][3] [0][4] [1][0] [1][1] [1][2] [1][3] [1][4] [2][0] [2][1] [2][2] [2][3] [2][4] [3][0] [3][1] [3][2] [3][3] [3][4] [4][0] [4][1] [4][2] [4][3] [4][4] %

chosunghwa@joseonghwaui-MacBookAir 4-1 % diff -w trans2-out.txt

chosunghwa@joseonghwaui-MacBookAir 4-1 %

(5)trans1

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix5test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out < trans1-in.txt > trans1-out.txt

n = A

[0][0] [0][1] [0][2] [0][3] [0][4] [1][0] [1][1] [1][2] [1][3] [1][4] [2][0] [2][1] [2][2] [2][3] [2][4] [3][0] [3][1] [3][2] [3][3] [3][4] [4][0] [4][1] [4][2] [4][3] [4][4] %

chosunghwa@joseonghwaui-MacBookAir 4-1 % diff -w trans1-out.txt trans1-exp.txt

chosunghwa@joseonghwaui-MacBookAir 4-1 %

(6) vector\_scan, vector\_print

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix6test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out

n = 3

x

[0][0]1

[1][0]2

[2][0]3

x

2.00000

4.00000

6.00000

chosunghwa@joseonghwaui-MacBookAir 4-1 %

(7)

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out

n = 3

A

[0][0] 1

[0][1] 1

[0][2] 1

[1][0] 1

[1][1] 2

[1][2] 3

[2][0] 5

[2][1] 4

[2][2] 3

x

[0][0]1

[1][0]2

[2][0]2

A

1.00000 1.00000 1.00000

1.00000 2.00000 3.00000

5.00000 4.00000 3.00000

x

1.00000

2.00000

2.00000

Y

5.00000

11.00000

19.00000

chosunghwa@joseonghwaui-MacBookAir 4-1 %

chosunghwa@joseonghwaui-MacBookAir 4-1 % gcc matrix.c matrix7test.c

chosunghwa@joseonghwaui-MacBookAir 4-1 % ./a.out < matrix7-in.txt > matrix7-out.txt

n = A

[0][0] [0][1] [0][2] [0][3] [0][4] [1][0] [1][1] [1][2] [1][3] [1][4] [2][0] [2][1] [2][2] [2][3] [2][4] [3][0] [3][1] [3][2] [3][3] [3][4] [4][0] [4][1] [4][2] [4][3] [4][4] x

[0][0][1][0][2][0][3][0][4][0]%

chosunghwa@joseonghwaui-MacBookAir 4-1 % diff -w matrix7-out.txt matrix7-exp.txt

chosunghwa@joseonghwaui-MacBookAir 4-1 %

感想：いろんな行列とベクトルの読み込み、プリント、計算を行う方法を学ぶことができました。

----------------------------------------

[課題 4.2]

結果：

(1) cool

void bmp\_cool(bmp\_header\_t \*hd, pixel\_t img[MAX\_Y][MAX\_X])

{

int i, j;

*/\*\*\*\* 以下 (現状は画像を全体に暗くする処理になっている) を書き換えて完成させよ \*\*\*\*/*

*for* (i=0; i< hd->biHeight; i++) { */\* biHeight は画像の高さ \*/*

*for* (j=0; j< hd->biWidth; j++) { */\* biWidth は画像の幅 \*/*

img[i][j].r = img[i][j].r\*0.8;

img[i][j].g = img[i][j].g\*1.0;

img[i][j].b = img[i][j].b\*1.0;

}

}

(2) mono

void bmp\_mono(bmp\_header\_t \*hd, pixel\_t img[MAX\_Y][MAX\_X])

{

int i, j;

*/\*\*\*\* 以下 (現状は画像を全体に暗くする処理になっている) を書き換えて完成させよ \*\*\*\*/*

*for* (i=0; i< hd->biHeight; i++) { */\* biHeight は画像の高さ \*/*

*for* (j=0; j< hd->biWidth; j++) { */\* biWidth は画像の幅 \*/*

img[i][j].r = img[i][j].g = img[i][j].b; *//mono はrgbの値が同じ*

}

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ここまで \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

}

(3) フレーム

*left axis*

*for* (i = 0; i < hd->biWidth; i++) {

*for* ( j = 0; j < hd->biHeight - 315 ; j++) {

img[i][j] = red;

}

}

*//top cross*

*for* (i = hd->biHeight - 5; i < hd->biHeight; i++) {

*for* (j = 0; j < hd->biWidth; j++) {

img[i][j] = red;

}

}

*//bottom cross*

*for* (i = 0; i < hd->biHeight - 315; i++) {

*for* (j = 0; j < hd->biWidth; j++) {

img[i][j] = red;

}

}

*//right axis*

*for* (i = 0; i < hd->biHeight; i++) {

*for* (j = hd->biWidth - 5; j < hd->biWidth; j++) {

img[i][j] = red;

}

}

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ここまで \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

(4)mirror

void bmp\_mirror(bmp\_header\_t \*hd, pixel\_t img[MAX\_Y][MAX\_X])

{

int i, j;

pixel\_t temp;

*for* (i = 0; i < hd->biHeight / 2; i++) {

*for* (j = 0; j < hd->biWidth; j++) {

*// 현재 픽셀과 대칭되는 픽셀을 스왑*

temp = img[i][j];

img[i][j] = img[hd->biHeight - 1 - i][j];

img[hd->biHeight - 1 - i][j] = temp;

}

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ここまで \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

}

(5)mosaic

void bmp\_mosaic(bmp\_header\_t \*hd, pixel\_t img[MAX\_Y][MAX\_X])

{

int i, j;

int width = hd->biWidth;

*// 이미지의 왼쪽 절반에만 모자이크 처리 적용*

*for* (i = 0; i < hd->biHeight; i += 16) {

*for* (j = 0; j < width / 2; j += 16) {

int sum\_r = 0, sum\_g = 0, sum\_b = 0;

int count = 0;

*// 블록 내의 픽셀들의 평균값 계산*

*for* (int k = i; k < i + 16; k++) {

*for* (int l = j; l < j + 16; l++) {

sum\_r += img[k][l].r;

sum\_g += img[k][l].g;

sum\_b += img[k][l].b;

count++;

}

}

*// 블록 내의 픽셀을 평균값으로 설정*

*for* (int k = i; k < i + 16; k++) {

*for* (int l = j; l < j + 16; l++) {

img[k][l].r = sum\_r / count;

img[k][l].g = sum\_g / count;

img[k][l].b = sum\_b / count;

}

}

}

}

}

(6)collapse

void bmp\_collapse(bmp\_header\_t \*hd, pixel\_t img[MAX\_Y][MAX\_X])

{

int i, j;

int new\_height = hd->biHeight / 2;

hd->biHeight = new\_height;

*for* (i = 0; i < new\_height; i++) {

*for* (j = 0; j < hd->biWidth; j++) {

*// 각 두 픽셀의 색상값을 평균하여 설정*

img[i][j].r = (img[i \* 2][j].r + img[i \* 2 + 1][j].r) / 2;

img[i][j].g = (img[i \* 2][j].g + img[i \* 2 + 1][j].g) / 2;

img[i][j].b = (img[i \* 2][j].b + img[i \* 2 + 1][j].b) / 2;

}

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ここまで \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

}

感想：

----------------------------------------

[課題 4.3]

結果:

Image2.c

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <assert.h>

*#include* "image2.h"

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

*24ビット-ビットマップデータ (BMPファイル) のファイル入出力*

*(制限)*

*- 画像の横幅は４の倍数であること*

*- リトルエンディアンにのみ対応*

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

void bmp\_read (

char filename[],

bmp\_data\_t \*bmp

)

{

FILE \*fp;

int i, j;

*/\* ファイルオープン \*/*

*if* ((fp = fopen(filename, "rb"))==NULL) {

fprintf(stderr, "read\_bmp: failed to open \"%s\"\n", filename);

exit(1);

}

fprintf(stderr, "input file = \"%s\"\n", filename);

*/\* ヘッダ情報の読み込み \*/*

fread(&bmp->header.bfType, sizeof(bmp->header.bfType), 1, fp);

fread(&bmp->header.bfSize, sizeof(bmp->header.bfSize), 1, fp);

fread(&bmp->header.bfReserved1, sizeof(bmp->header.bfReserved1), 1, fp);

fread(&bmp->header.bfReserved2, sizeof(bmp->header.bfReserved2), 1, fp);

fread(&bmp->header.bfOffBits, sizeof(bmp->header.bfOffBits), 1, fp);

fread(&bmp->header.biSize, sizeof(bmp->header.biSize), 1, fp);

fread(&bmp->header.biWidth, sizeof(bmp->header.biWidth), 1, fp);

fread(&bmp->header.biHeight, sizeof(bmp->header.biHeight), 1, fp);

fread(&bmp->header.biPlanes, sizeof(bmp->header.biPlanes), 1, fp);

fread(&bmp->header.biBitCount, sizeof(bmp->header.biBitCount), 1, fp);

fread(&bmp->header.biCompression, sizeof(bmp->header.biCompression), 1, fp);

fread(&bmp->header.biSizeImage, sizeof(bmp->header.biSizeImage), 1, fp);

fread(&bmp->header.biXPelsPerMeter, sizeof(bmp->header.biXPelsPerMeter), 1, fp);

fread(&bmp->header.biYPelsPerMeter, sizeof(bmp->header.biYPelsPerMeter), 1, fp);

fread(&bmp->header.biClrUsed, sizeof(bmp->header.biClrUsed), 1, fp);

fread(&bmp->header.biClrImportant, sizeof(bmp->header.biClrImportant), 1, fp);

assert(bmp->header.biHeight<=MAX\_Y);

assert(bmp->header.biWidth<=MAX\_X);

*/\* ビットマップデータの読み込み \*/*

*for* (i=0; i < bmp->header.biHeight; i++) {

*for* (j=0; j < bmp->header.biWidth; j++) {

fread(&bmp->img[i][j].b, 1, 1, fp);

fread(&bmp->img[i][j].g, 1, 1, fp);

fread(&bmp->img[i][j].r, 1, 1, fp);

}

}

*/\* ファイルクローズ \*/*

fclose(fp);

}

void bmp\_write(

char filename[],

bmp\_data\_t \*bmp

)

{

FILE \*fp;

int i, j;

assert(bmp->header.biHeight<=MAX\_Y);

assert(bmp->header.biWidth<=MAX\_X);

*/\* ファイルオープン \*/*

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

fprintf(stderr, "write\_bmp: failed to open \"%s\"\n", filename);

exit(1);

}

fprintf(stderr, "output file = %s\n", filename);

*/\* ヘッダ情報の書き出し \*/*

fwrite(&bmp->header.bfType, sizeof(bmp->header.bfType), 1, fp);

fwrite(&bmp->header.bfSize, sizeof(bmp->header.bfSize), 1, fp);

fwrite(&bmp->header.bfReserved1, sizeof(bmp->header.bfReserved1), 1, fp);

fwrite(&bmp->header.bfReserved2, sizeof(bmp->header.bfReserved2), 1, fp);

fwrite(&bmp->header.bfOffBits, sizeof(bmp->header.bfOffBits), 1, fp);

fwrite(&bmp->header.biSize, sizeof(bmp->header.biSize), 1, fp);

fwrite(&bmp->header.biWidth, sizeof(bmp->header.biWidth), 1, fp);

fwrite(&bmp->header.biHeight, sizeof(bmp->header.biHeight), 1, fp);

fwrite(&bmp->header.biPlanes, sizeof(bmp->header.biPlanes), 1, fp);

fwrite(&bmp->header.biBitCount, sizeof(bmp->header.biBitCount), 1, fp);

fwrite(&bmp->header.biCompression, sizeof(bmp->header.biCompression), 1, fp);

fwrite(&bmp->header.biSizeImage, sizeof(bmp->header.biSizeImage), 1, fp);

fwrite(&bmp->header.biXPelsPerMeter, sizeof(bmp->header.biXPelsPerMeter), 1, fp);

fwrite(&bmp->header.biYPelsPerMeter, sizeof(bmp->header.biYPelsPerMeter), 1, fp);

fwrite(&bmp->header.biClrUsed, sizeof(bmp->header.biClrUsed), 1, fp);

fwrite(&bmp->header.biClrImportant, sizeof(bmp->header.biClrImportant), 1, fp);

*/\* ビットマップデータの書き出し \*/*

*for* (i=0; i < bmp->header.biHeight; i++) {

*for* (j=0; j < bmp->header.biWidth; j++) {

fwrite(&bmp->img[i][j].b, 1, 1, fp);

fwrite(&bmp->img[i][j].g, 1, 1, fp);

fwrite(&bmp->img[i][j].r, 1, 1, fp);

}

}

*/\* ファイルクローズ \*/*

fclose(fp);

}

Image2collapse.c

*#include* <stdio.h>

*#include* "image2.h"

*/\* 画像全体を「涼しげ」にする関数 \*/*

void bmp\_collapse(bmp\_data\_t \*bmp);

int main(void)

{

bmp\_data\_t bmp;

bmp\_read("in.bmp", &bmp); */\* in.bmp から読み込む \*/*

bmp\_collapse(&bmp); */\* 処理の呼び出し \*/*

bmp\_write("out.bmp", &bmp); */\* out.bmp への書き出し \*/*

*return* 0;

}

*/\* 画像処理の関数の本体 \*/*

void bmp\_collapse(bmp\_data\_t \*bmp)

{

int i, j;

int new\_height = bmp->header.biHeight / 2;

bmp->header.biHeight = new\_height;

*for* (i = 0; i < bmp->header.biHeight; i++) {

*for* (j = 0; j < bmp->header.biWidth; j++) {

*// 각 두 픽셀의 색상값을 평균하여 설정*

bmp->img[i][j].r = (bmp->img[i \* 2][j].r + bmp->img[i \* 2 + 1][j].r) / 2;

bmp->img[i][j].g = (bmp->img[i \* 2][j].g + bmp->img[i \* 2 + 1][j].g) / 2;

bmp->img[i][j].b = (bmp->img[i \* 2][j].b + bmp->img[i \* 2 + 1][j].b) / 2;

}

}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ここまで \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

}

感想：

image2.hでは構造体の中に構造体があって、構造体の中の構造体を呼び出す方法を学ぶことができました。

----------------------------------------

[課題 B4 の感想]

2次元配列の計算、呼び出し、などを学び、配列の計算をすることだできるようになりました。そして4-2で、画像処理を通じて配列の利用、構造体、ポインタを全部利用して画像することだ面白かったです。