

[演習 08] 標本化と量子化 (2)

学生番号 : 20216187

氏名 : 劉潤之

提出日 : 2022/12/01

[レポート作成の準備]

1. script コマンドを用いて、プログラムリスト・コンパイル結果・実行結果を一つのファイルに書き出しておく。(演習問題ごとに、ファイルを書き出しておくこと.)
2. レポートに画像を載せる必要がある場合には、画像を準備しておく。(どのような図を記載すべきかについては問題文に示されている.)

[レポート作成方法]

次ページ以降において、演習問題ごとに、以下を実施すること。

1. 問題番号を、ドロップダウンリストより 2 箇所選択する。
2. プログラムリスト・コンパイル結果・実行結果を、準備しておいたファイルからコピーし、所定の場所に、テキスト形式でペーストする。
3. 必要に応じて、所定の場所に図を挿入するとともに、その題目や説明を記入する。

なお、未使用なページやスペースは、そのままにしておいてよい(削除しなくてよい)。

[レポート提出方法]

すべての演習問題を終了した後、レポートを作成して提出する。ここで、レポートの提出期限は、原則として、次回演習日の前夜までとする。

提出は、工学部の「ポータルサイト」の課題提出のページから実施すること。なお、提出の際、コメントを特に記載する必要はない。

[提出ファイル]

提出は、PDF ファイルとする。(word でレポートを作成し、一旦保存する。その後、同ファイルを「名前を付けて保存(コピーを保存)」する。ここで、“ファイル名”の下にある“ファイルの種類”を PDF とすれば、PDF ファイルが作成できる。)

なお、ファイル名は、“01_xxxxxxx.pdf”とする。ここで、先頭の 2 桁の値は演習問題の回、末尾の xxxxxxxx の部分は学生番号とする。

【問題 8-A-1】

[プログラムリスト・コンパイル結果・実行結果]

```
Script started on Thu Dec  1 17:29:06 2022
u20216187@gw[31]: cat e x_08_A _01.c
```

```
#define _CRT_SECURE_NO_WARNINGS
```

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

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

```
#include<math.h>
```

```
#define MAXLENGTH 100
```

```
unsigned char header[54];
```

```
unsigned char imgin[3][512][512];
```

```
unsigned char imgout[3][512][512];
```

```
double dtemp[3][512][512];
```

```
int itemp[3][512][512];
```

```
void convert(int n);
```

```
void get_data();
```

```
void rgb_to_ybr();
```

```
void processing();
```

```
void ybr_to_rgb();
```

```
void put_data();
```

```
int calculate(int a, int b);
```

```
int width, height, bite;
```

```
double rgb_con_ybr[3][3] = {
    { 0.2990, 0.5870, 0.1140},
```

```
{-0.1687,-0.3313, 0.5000},
{ 0.5000,-0.4187,-0.0813}
};

double ybr_con_rgb[3][3] = {
    { 1.0000, 0.0000, 1.4020},
    { 1.0000,-0.3441,-0.7141},
    { 1.0000, 1.7720, 0.0000}
};

//char ybr_name[3][3] = { "Y", "Cb", "Cr"};

int main(){

    get_data();
    rgb_to_ybr();
    processing();
    ybr_to_rgb();
    put_data();

    return 0;
}

void get_data(){
    FILE *fp;

    char filename[MAXLENGTH];

    int i, c, x, y;

    int filesize, offset, bite_pixel;
```

```
printf("ファイル名を入力して下さい:");  
  
scanf("%s", filename);  
  
fp = fopen(filename, "rb");  
  
if(fp == NULL){  
    printf("ファイルをオープンできません.¥n");  
    exit(1);  
}  
  
printf("ファイルをオープンしました.¥n");  
  
for(i = 0; i < 54; i ++){  
    c = fgetc(fp);  
    header[i] = (unsigned char)c;  
}  
  
/*  //-----  
printf("¥n<ファイルタイプ>¥n");  
printf("header[0]=");  
convert(header[0]);  
  
printf(" header[1]=");  
convert(header[1]);  
printf("¥n");  
*/  
//-----
```

```
printf("¥n<ファイルサイズ>¥n");

printf("header[2]=");
convert(header[2]);

printf(" header[3]=");
convert(header[3]);

printf(" header[4]=");
convert(header[4]);

printf(" header[5]=");
convert(header[5]);

filesize = calculate(5, 2);
printf("¥n%d バイト¥n", filesize);

/*//-----

printf("¥n<予約領域>¥n");

printf("header[6]=");
convert(header[6]);

printf(" header[7]=");
convert(header[7]);

printf(" header[8]=");
convert(header[8]);

printf(" header[9]=");
convert(header[9]);
```

```
printf("¥n");

*/

//-----

printf("¥n<オフセット>¥n");

printf("header[10]=");

convert(header[10]);


printf(" header[11]=");

convert(header[11]);


printf(" header[12]=");

convert(header[12]);


printf(" header[13]=");

convert(header[13]);


offset = calculate(13, 10),
printf("¥n%d バイト¥n", offset);

/*//-----

printf("¥n<情報ベッタサイズ>¥n");

printf("header[14]=");

convert(header[14]);


printf(" header[15]=");

convert(header[15]);


printf(" header[16]=");

convert(header[16]);
```

```
printf(" header[17]=");
convert(header[17]);
printf("¥n");
*/
//-----
printf("¥n<画像の幅>¥n");
printf("header[18]=");
convert(header[18]);

printf(" header[19]=");
convert(header[19]);

printf(" header[20]=");
convert(header[20]);

printf(" header[21]=");
convert(header[21]);

width = calculate(21, 18);
printf("¥n%d 画素¥n",width);
//-----
printf("¥n<画像の高さ>¥n");
printf("header[22]=");
convert(header[22]);

printf(" header[23]=");
convert(header[23]);
```

```
printf(" header[24]=");
convert(header[24]);

printf(" header[25]=");
convert(header[25]);

height = calculate(25, 22);
printf("¥n%d ライン¥n",height);
/*//-----
printf("¥n<色プレーン数>¥n");
printf("header[26]=");
convert(header[26]);

printf(" header[27]=");
convert(header[27]);
printf("¥n");*/
//-----
printf("¥n<1 画素当たりのビット数>¥n");
printf("header[28]=");
convert(header[28]);

printf(" header[29]=");
convert(header[29]);

bite_pixel = calculate(29, 28);
printf("¥n%d ビット¥n",bite_pixel);
/*//-----
printf("¥n<圧縮方式>¥n");
printf("header[30]=");
```



```
convert(header[30]);

printf(" header[31]=");
convert(header[31]);

printf(" header[32]=");
convert(header[32]);

printf(" header[33]=");
convert(header[33]);
printf("¥n");
//-----

printf("¥n<画像データサイズ>¥n");
printf("header[34]=");
convert(header[34]);

printf(" header[35]=");
convert(header[35]);

printf(" header[36]=");
convert(header[36]);

printf(" header[37]=");
convert(header[37]);
printf("¥n");
//-----

printf("¥n<水平解像度>¥n");
printf("header[38]=");
```

```
convert(header[38]);

printf(" header[39]=");
convert(header[39]);

printf(" header[40]=");
convert(header[40]);

printf(" header[41]=");
convert(header[41]);
printf("¥n");
//-----

printf("¥n<垂直解像度>¥n");
printf("header[42]=");
convert(header[42]);

printf(" header[43]=");
convert(header[43]);

printf(" header[44]=");
convert(header[44]);

printf(" header[45]=");
convert(header[45]);
printf("¥n");
//-----

printf("¥n<色数>¥n");
printf("header[46]=");
convert(header[46]);
```

```
printf(" header[47]=");
convert(header[47]);

printf(" header[48]=");
convert(header[48]);

printf(" header[49]=");
convert(header[49]);
printf("¥n");
//-----

printf("¥n<重要な色数>¥n");
printf("header[50]=");
convert(header[50]);

printf(" header[51]=");
convert(header[51]);

printf(" header[52]=");
convert(header[52]);

printf(" header[53]=");
convert(header[53]);
printf("¥n");
*/

//-----

printf("¥n<挿入ビット数>¥n");
bite = (offset + width * height * (bite_pixel / 8)) % 4;
```

```
printf("%d バイト¥n",bite);

for(y = height - 1; y >= 0; y--){
    for(x = 0; x < width; x++){
        for(i = 2; i >= 0; i--){
            c = fgetc(fp);
            imgin[i][x][y] = (unsigned char)c;
        }
    }
}

fclose(fp);
printf("¥n%s をクローズしました.¥n", filename);

}

void convert(int n){

    printf("%02x", n);

}

int calculate(int a, int b){
    int i;
    int value;
    value = header[a];
    for(i = a-1; i >= b; i--){
        value <= 8;
        value += header[i];
    }
}
```

```
    }

    return value;
}

void processing(){

    //int i, x, y;

    ///int copy[3];

    //int block, b_height, b_width;

    //block = 2;

    //b_height = height / block;

    //b_width = width / block;

    //for(y = 0; y < height; y++){
    //  for(x = 0; x < width; x++){
    //    for(i = 0; i < 3; i++){
    //      if((x / b_width + y / b_height) % 2 == 0){
    //        imgout[i][x][y] = imgin[i][x][y];
    //      }else if(i != 0){
    //        imgout[i][x][y] = 128;
    //      }else{
    //        imgout[i][x][y] = 0;
    //      }
    //    }
    //  }
    // }

    //}

    /*

    printf("¥n コピーモードを入力して下さい.¥n");
```

```
printf("(コピーする場合 : 1, 固定値に置き換える場合 : 0)¥n");

for (i = 0; i < 3; i++){

    printf("%-2s : ", ybr_name[i]);

    scanf("%d", &copy[i]);

}

for(i = 0; i < 3; i++){

    if (copy[i] == 1){

        for(y = 0; y < height; y++){

            for(x = 0; x < width; x++){

                imgout[i][x][y] = imgin[i][x][y];

            }

        }

    }else{

        for(y = 0; y < height; y++){

            for(x = 0; x < width; x++){

                imgout[i][x][y] = 128;

            }

        }

    }

}

*/
```

```
/*
```

```
for(y = 0; y < height; y++){

    for(x = 0; x < width; x++){
```

```
    for(i = 0; i < 3; i++){
        imgout[i][x][y] = imgin[i][x][y];
    }
}
}
*/

//int i, x, y;

//int copy[3];

// int block/*, b_height, b_width*/;


//int xi, yi;

//int ave,sum;

/* while(1){
    printf("ブロックサイズを入力してください(2,4,8,16)¥n");
    scanf("%d",&block);

    double id , cal;

    id= log2(block);

    cal =(double) id -(int)id;

    if(cal ==0 && block<=16 && block>0){
break;

    } else {
printf("入力ミス、再入力してください¥n");

    }
}

for (y = 0; y < height; y += block) {
    for (x = 0; x < width; x += block) {

        sum = 0;
```

```
    for (yi = 0; yi < block; yi++)
        for (xi = 0; xi < block; xi++)
            sum += imgin[0][x + xi][y + yi];

ave = (int)((double)sum / (double)(block * block)+0.5);

if (ave > 255) {
    ave = 255;
}
else if (ave < 0) {
    ave = 0;
}

for (yi = 0; yi < block; yi++)
    for (xi = 0; xi < block; xi++)
        imgout[0][x + xi][y + yi] = (unsigned char)ave;

}

}

for (i = 1; i < 3; i++) {
    for (y = 0; y < height; y++) {
        for (x = 0; x < width; x++) {

            imgout[i][x][y] = 128;
```



```
        }
    }
}*/

int x,y,i;
for(y=0;y<height;y++){
    for(x=0;x<width;x++){
        for(i=2;i>=0;i--){
            if(i==0){
                if(0<=imgin[i][x][y]&&imgin[i][x][y]<128){
                    imgin[i][x][y]=64;
                }else if(128<=imgin[i][x][y]&&imgin[i][x][y]<=255){
                    imgin[i][x][y]=192;
                }
            }
        }
        else
        {
            imgin[i][x][y]=128;
        }
        imgout[i][x][y]=imgin[i][x][y];
    }
}

printf("入力画像データをコピーして出力画像データを作成しました.\n");
}
```

```
void put_data(){  
    FILE *fp;  
  
    char filename1[MAXLENGTH];  
  
    int i,x,y;  
  
    printf("出力ファイル名を入力して下さい:");  
  
    scanf("%s",filename1);  
  
    fp = fopen(filename1,"w");  
  
    printf("%s をオープンしました.¥n", filename1);  
  
    for (i = 0; i < 54; i++){  
        fputc(header[i], fp);  
    }  
  
    for(y = height - 1; y >= 0; y--){  
        for(x = 0; x < width; x++){  
            for(i = 2;i >= 0; i--){  
                fputc(imgout[i][x][y], fp);  
            }  
        }  
    }  
  
    for(i = 0; i < bite; i++){  
        fputc('¥0', fp);  
    }  
}
```

```
fclose(fp);

printf("%s をクローズしました.\n", filename1);

}

void rgb_to_ybr(){
    int i, x, y, j;

    /*
    printf("\n<入力 RGB 信号(整数値)>\n");

    if(height <= 16 || width <= 16){

        printf("\n<R 信号>\n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
                printf("%02x ", imgin[0][x][y]);
            }
            printf("\n");
        }

        printf("\n<G 信号>\n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
                printf("%02x ", imgin[1][x][y]);
            }
            printf("\n");
        }
    }
}
```

```
}

printf("¥n<B 信号>¥n");

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgin[2][x][y]);
    }
    printf("¥n");
}

}else{

    printf("画像サイズが大きいため表示しません.¥n");
}

*/

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){
            dtemp[i][x][y] = 0.0;

            for (j = 0; j < 3; j++)
                dtemp[i][x][y] += rgb_con_ybr[i][j] * (double)imgin[j][x][y];
        }
    }
}

}

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){

            if(dtemp[i][x][y] > 0.0){
```

```
        itemp[i][x][y] = (int)(dtemp[i][x][y] + 0.5);
    }else{
        itemp[i][x][y] = (int)(dtemp[i][x][y] - 0.5);
    }

    if (i != 0){
        itemp[i][x][y] += 128;
    }

    if (itemp[i][x][y] > 255){
        itemp[i][x][y] = 255;
    }else if (itemp[i][x][y] < 0){
        itemp[i][x][y] = 0;
    }

    imgin[i][x][y] = (unsigned char)itemp[i][x][y];
}
}
}

/*
printf("¥n<入力 YCbCr 信号(整数値)>¥n");

if(height <= 16 || width <= 16){

    printf("¥n<Y 信号>¥n");
    for(y = 0; y < height; y++){
        for(x = 0; x < width; x++){
            printf("%02x ", imgin[0][x][y]);
```

```
    }

    printf("¥n");
}

printf("¥n<Cb 信号>¥n");
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgin[1][x][y]);
    }
    printf("¥n");
}

printf("¥n<Cr 信号>¥n");
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgin[2][x][y]);
    }
    printf("¥n");
}
}else{
    printf("画像サイズが大きいため表示しません.¥n");
}
*/
}

void ybr_to_rgb(){
    int i, x, y, j;
```

```
/*  
  
printf("¥n<入力 YCbCr 信号(整数値)>¥n");  
  
if(height <= 16 || width <= 16){  
  
    printf("¥n<Y 信号>¥n");  
    for(y = 0; y < height; y++){  
        for(x = 0; x < width; x++){  
            printf("%02x ", imgin[0][x][y]);  
        }  
        printf("¥n");  
    }  
  
    printf("¥n<Cb 信号>¥n");  
    for(y = 0; y < height; y++){  
        for(x = 0; x < width; x++){  
            printf("%02x ", imgin[1][x][y]);  
        }  
        printf("¥n");  
    }  
  
    printf("¥n<Cr 信号>¥n");  
    for(y = 0; y < height; y++){  
        for(x = 0; x < width; x++){  
            printf("%02x ", imgin[2][x][y]);  
        }  
        printf("¥n");  
    }  
}  
}else{
```

```
printf("画像サイズが大きいため表示しません.¥n");
}
*/

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){
            dtemp[i][x][y] = 0.0;
            for (j = 0; j < 3; j++)
                if (j == 0)
                    dtemp[i][x][y] += ybr_con_rgb[i][j] * (double)imgout[j][x][y];
                else
                    dtemp[i][x][y] += ybr_con_rgb[i][j] * (double)(imgout[j][x][y] -
128);
        }
    }
}

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){

            if(dtemp[i][x][y] > 0.0){
                itemp[i][x][y] = (int)(dtemp[i][x][y] + 0.5);
            }else{
                itemp[i][x][y] = (int)(dtemp[i][x][y] - 0.5);
            }
        }
    }
}
```



```
    if (itemp[i][x][y] > 255){
        itemp[i][x][y] = 255;
    }else if (itemp[i][x][y] < 0){
        itemp[i][x][y] = 0;
    }

    imgout[i][x][y] = (unsigned char)itemp[i][x][y];
}
}
}

/*
printf("¥n<入力 RGB 信号(整数値)>¥n");

if(height <= 16 || width <= 16){

    printf("¥n<R 信号>¥n");
    for(y = 0; y < height; y++){
        for(x = 0; x < width; x++){
            printf("%02x ", imgout[0][x][y]);
        }
        printf("¥n");
    }

    printf("¥n<G 信号>¥n");
    for(y = 0; y < height; y++){
        for(x = 0; x < width; x++){
            printf("%02x ",imgout[1][x][y]);
```

```

    }

    printf("%n");
}

printf("%n<B 信号>%n");
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgout[2][x][y]);
    }
    printf("%n");
}
}else{
    printf("画像サイズが大きいため表示しません.%n");
}
*/
}

u20216187@gw[32]: gcc -Wall e x_08_A _01.c -o e x_08_A _01
u20216187@gw[33]: ./e x_08_A_01

ファイル名を入力して下さい:@ 0pepper.bmp
ファイルをオープンしました.

<ファイルサイズ>
header[2]=38 header[3]=00 header[4]=03 header[5]=00
196664 バイト

<オフセット>
header[10]=36 header[11]=00 header[12]=00 header[13]=00
54 バイト

<画像の幅>
header[18]=00 header[19]=01 header[20]=00 header[21]=00
256 画素

<画像の高さ>
header[22]=00 header[23]=01 header[24]=00 header[25]=00
256 ライン

<1 画素当たりのビット数>
header[28]=18 header[29]=00

```

24 ビット

<挿入ビット数>

2 バイト

```
pepper.bmp をクローズしました.  
入力画像データをコピーして出力画像データを作成しました.  
出力ファイル名を入力して下さい:pepper_08_A_1.bmp  
pepper_08_A_1.bmp をオープンしました.  
pepper_08_A_1.bmp をクローズしました.  
u20216187@gw[34]: exit
```

```
exit
```

```
Script done on Thu Dec 1 17:30:16 2022
```

[添付図]



ここをクリックまたはタップしてテキスト
を入力してください。



ここをクリックまたはタップしてテキスト
を入力してください。

【問題 8-B-1】

[プログラムリスト・コンパイル結果・実行結果]

```
Script started on Thu Dec  1 17:30:23 2022
u20216187@gw[31]: cat e x_08_B _01.c
```

```
#define _CRT_SECURE_NO_WARNINGS
```

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

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

```
#include<math.h>
```

```
#define MAXLENGTH 100
```

```
unsigned char header[54];
```

```
unsigned char imgin[3][512][512];
```

```
unsigned char imgout[3][512][512];
```

```
double dtemp[3][512][512];
```

```
int itemp[3][512][512];
```

```
void convert(int n);
```

```
void get_data();
```

```
void rgb_to_ybr();
```

```
void processing();
```

```
void ybr_to_rgb();
```

```
void put_data();
```

```
int calculate(int a, int b);
```

```
int width, height, bite;
```

```
double rgb_con_ybr[3][3] = {
    { 0.2990, 0.5870, 0.1140},
```

```
{-0.1687,-0.3313, 0.5000},
{ 0.5000,-0.4187,-0.0813}
};

double ybr_con_rgb[3][3] = {
    { 1.0000, 0.0000, 1.4020},
    { 1.0000,-0.3441,-0.7141},
    { 1.0000, 1.7720, 0.0000}
};

//char ybr_name[3][3] = { "Y", "Cb", "Cr"};

int main(){

    get_data();
    rgb_to_ybr();
    processing();
    ybr_to_rgb();
    put_data();

    return 0;
}

void get_data(){
    FILE *fp;

    char filename[MAXLENGTH];

    int i, c, x, y;

    int filesize, offset, bite_pixel;
```

```
printf("ファイル名を入力して下さい:");  
  
scanf("%s", filename);  
  
fp = fopen(filename, "rb");  
  
if(fp == NULL){  
    printf("ファイルをオープンできません.¥n");  
    exit(1);  
}  
  
printf("ファイルをオープンしました.¥n");  
  
for(i = 0; i < 54; i ++){  
    c = fgetc(fp);  
    header[i] = (unsigned char)c;  
}  
  
/*  //-----  
printf("¥n<ファイルタイプ>¥n");  
printf("header[0]=");  
convert(header[0]);  
  
printf(" header[1]=");  
convert(header[1]);  
printf("¥n");  
*/  
//-----
```

```
printf("¥n<ファイルサイズ>¥n");

printf("header[2]=");
convert(header[2]);

printf(" header[3]=");
convert(header[3]);

printf(" header[4]=");
convert(header[4]);

printf(" header[5]=");
convert(header[5]);

filesize = calculate(5, 2);
printf("¥n%d バイト¥n", filesize);
/*//-----

printf("¥n<予約領域>¥n");
printf("header[6]=");
convert(header[6]);

printf(" header[7]=");
convert(header[7]);

printf(" header[8]=");
convert(header[8]);

printf(" header[9]=");
convert(header[9]);
```

```
printf("¥n");

*/

//-----

printf("¥n<オフセット>¥n");

printf("header[10]=");

convert(header[10]);


printf(" header[11]=");

convert(header[11]);


printf(" header[12]=");

convert(header[12]);


printf(" header[13]=");

convert(header[13]);


offset = calculate(13, 10),
printf("¥n%d バイト¥n", offset);

/*//-----

printf("¥n<情報ベッタサイズ>¥n");

printf("header[14]=");

convert(header[14]);


printf(" header[15]=");

convert(header[15]);


printf(" header[16]=");

convert(header[16]);
```



```
printf(" header[17]=");
convert(header[17]);
printf("¥n");
*/
//-----
printf("¥n<画像の幅>¥n");
printf("header[18]=");
convert(header[18]);

printf(" header[19]=");
convert(header[19]);

printf(" header[20]=");
convert(header[20]);

printf(" header[21]=");
convert(header[21]);

width = calculate(21, 18);
printf("¥n%d 画素¥n",width);
//-----
printf("¥n<画像の高さ>¥n");
printf("header[22]=");
convert(header[22]);

printf(" header[23]=");
convert(header[23]);
```

```
printf(" header[24]=");
convert(header[24]);

printf(" header[25]=");
convert(header[25]);

height = calculate(25, 22);
printf("¥n%d ライン¥n",height);
/*//-----
printf("¥n<色プレーン数>¥n");
printf("header[26]=");
convert(header[26]);

printf(" header[27]=");
convert(header[27]);
printf("¥n");*/
//-----
printf("¥n<1 画素当たりのビット数>¥n");
printf("header[28]=");
convert(header[28]);

printf(" header[29]=");
convert(header[29]);

bite_pixel = calculate(29, 28);
printf("¥n%d ビット¥n",bite_pixel);
/*//-----
printf("¥n<圧縮方式>¥n");
printf("header[30]=");
```

```
convert(header[30]);

printf(" header[31]=");
convert(header[31]);

printf(" header[32]=");
convert(header[32]);

printf(" header[33]=");
convert(header[33]);
printf("¥n");
//-----

printf("¥n<画像データサイズ>¥n");
printf("header[34]=");
convert(header[34]);

printf(" header[35]=");
convert(header[35]);

printf(" header[36]=");
convert(header[36]);

printf(" header[37]=");
convert(header[37]);
printf("¥n");
//-----

printf("¥n<水平解像度>¥n");
printf("header[38]=");
```

```
convert(header[38]);

printf(" header[39]=");
convert(header[39]);

printf(" header[40]=");
convert(header[40]);

printf(" header[41]=");
convert(header[41]);
printf("¥n");
//-----

printf("¥n<垂直解像度>¥n");
printf("header[42]=");
convert(header[42]);

printf(" header[43]=");
convert(header[43]);

printf(" header[44]=");
convert(header[44]);

printf(" header[45]=");
convert(header[45]);
printf("¥n");
//-----

printf("¥n<色数>¥n");
printf("header[46]=");
convert(header[46]);
```

```
printf(" header[47]=");
convert(header[47]);

printf(" header[48]=");
convert(header[48]);

printf(" header[49]=");
convert(header[49]);
printf("¥n");
//-----

printf("¥n<重要な色数>¥n");
printf("header[50]=");
convert(header[50]);

printf(" header[51]=");
convert(header[51]);

printf(" header[52]=");
convert(header[52]);

printf(" header[53]=");
convert(header[53]);
printf("¥n");
*/

//-----

printf("¥n<挿入ビット数>¥n");
bite = (offset + width * height * (bite_pixel / 8)) % 4;
```

```
printf("%d バイト¥n",bite);

for(y = height - 1; y >= 0; y--){
    for(x = 0; x < width; x++){
        for(i = 2; i >= 0; i--){
            c = fgetc(fp);
            imgin[i][x][y] = (unsigned char)c;
        }
    }
}

fclose(fp);
printf("¥n%s をクローズしました.¥n", filename);

}

void convert(int n){

    printf("%02x", n);

}

int calculate(int a, int b){
    int i;
    int value;
    value = header[a];
    for(i = a-1; i >= b; i--){
        value <= 8;
        value += header[i];
    }
}
```

```
}

return value;

}

void processing(){

    //int i, x, y;

    ///int copy[3];

    //int block, b_height, b_width;

    //block = 2;

    //b_height = height / block;

    //b_width = width / block;

    //for(y = 0; y < height; y++){
    //    for(x = 0; x < width; x++){
    //        for(i = 0; i < 3; i++){
    //            if((x / b_width + y / b_height) % 2 == 0){
    //                imgout[i][x][y] = imgin[i][x][y];
    //            }else if(i != 0){
    //                imgout[i][x][y] = 128;
    //            }else{
    //                imgout[i][x][y] = 0;
    //            }
    //        }
    //    }
    // }

    //}

    /*

    printf("¥n コピーモードを入力して下さい.¥n");
```

```
printf("(コピーする場合 : 1, 固定値に置き換える場合 : 0)¥n");

for (i = 0; i < 3; i++){

    printf("%-2s : ", ybr_name[i]);

    scanf("%d", &copy[i]);

}

for(i = 0; i < 3; i++){

    if (copy[i] == 1){

        for(y = 0; y < height; y++){

            for(x = 0; x < width; x++){

                imgout[i][x][y] = imgin[i][x][y];

            }

        }

    }else{

        for(y = 0; y < height; y++){

            for(x = 0; x < width; x++){

                imgout[i][x][y] = 128;

            }

        }

    }

}

*/
```

```
/*
```

```
for(y = 0; y < height; y++){

    for(x = 0; x < width; x++){
```



```
    for(i = 0; i < 3; i++){
        imgout[i][x][y] = imgin[i][x][y];
    }
}
}
*/

//int i, x, y;

//int copy[3];

// int block/*, b_height, b_width*/;


//int xi, yi;

//int ave,sum;

/* while(1){
    printf("ブロックサイズを入力してください(2,4,8,16)¥n");
    scanf("%d",&block);
    double id , cal;
    id= log2(block);
    cal =(double) id -(int)id;
    if(cal ==0 && block<=16 && block>0){
break;
    } else {
printf("入力ミス、再入力してください¥n");
    }
}

for (y = 0; y < height; y += block) {
    for (x = 0; x < width; x += block) {

        sum = 0;
```

```
    for (yi = 0; yi < block; yi++)
        for (xi = 0; xi < block; xi++)
            sum += imgin[0][x + xi][y + yi];

ave = (int)((double)sum / (double)(block * block)+0.5);

if (ave > 255) {
    ave = 255;
}
else if (ave < 0) {
    ave = 0;
}

for (yi = 0; yi < block; yi++)
    for (xi = 0; xi < block; xi++)
        imgout[0][x + xi][y + yi] = (unsigned char)ave;

}

}

for (i = 1; i < 3; i++) {
    for (y = 0; y < height; y++) {
        for (x = 0; x < width; x++) {

            imgout[i][x][y] = 128;
```

```
        }
    }
}*/

int i, x, y, n;

int block;

double id, cal;

while (1) {

    printf("¥n 出力画像の量子化レベル数を入力して下さい(2,4,8,16):");

    scanf("%d", &block);

    id = log2(block);
    cal = (double)id - (int)id;

    if (cal == 0 && block <= 16 && block > 0) {

        break;

    }

    else {

        printf("入力ミス、再入力してください¥n");

    }

}

n = 256 /block;

for ( i = 0; i <=2; i++) {

    for (y = 0; y < height; y++) {

        for (x = 0; x < width; x++) {

            if (i == 0) { imgout[i][x][y] = (imgin[i][x][y] / n) * n + n / 2; }

            else { imgout[i][x][y] = 128; }

        }

    }

}
```

```
}
```

```
printf("入力画像データをコピーして出力画像データを作成しました.¥n");
```

```
}
```

```
void put_data(){
```

```
FILE *fp;
```

```
char filename1[MAXLENGTH];
```

```
int i,x,y;
```

```
printf("出力ファイル名を入力して下さい:");
```

```
scanf("%s",filename1);
```

```
fp = fopen(filename1,"w");
```

```
printf("%s をオープンしました.¥n", filename1);
```

```
for (i = 0; i < 54; i++){
```

```
    fputc(header[i], fp);
```

```
}
```

```
for(y = height - 1; y >= 0; y--){
```

```
    for(x = 0; x < width; x++){
```

```
        for(i = 2;i >= 0; i--){
```

```
            fputc(imgout[i][x][y], fp);
```

```
        }
```

```
    }
```

```
}

for(i = 0; i < bite; i++){
    fputc('%0', fp);
}

fclose(fp);

printf("%s をクローズしました.\n", filename1);

}

void rgb_to_ybr(){
    int i, x, y, j;

    /*
    printf("\n<入力 RGB 信号(整数値)>\n");

    if(height <= 16 || width <= 16){

        printf("\n<R 信号>\n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
                printf("%02x ", imgin[0][x][y]);
            }
            printf("\n");
        }

        printf("\n<G 信号>\n");
```

```
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgin[1][x][y]);
    }
    printf("\n");
}

printf("\n<B 信号>\n");
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        printf("%02x ",imgin[2][x][y]);
    }
    printf("\n");
}
}else{
    printf("画像サイズが大きいため表示しません.\n");
}
*/

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){
            dtemp[i][x][y] = 0.0;
            for (j = 0; j < 3; j++){
                dtemp[i][x][y] += rgb_con_ybr[i][j] * (double)imgin[j][x][y];
            }
        }
    }
}
```

```
for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){

            if(dtemp[i][x][y] > 0.0){
                itemp[i][x][y] = (int)(dtemp[i][x][y] + 0.5);
            }else{
                itemp[i][x][y] = (int)(dtemp[i][x][y] - 0.5);
            }

            if (i != 0){
                itemp[i][x][y] += 128;
            }

            if (itemp[i][x][y] > 255){
                itemp[i][x][y] = 255;
            }else if (itemp[i][x][y] < 0){
                itemp[i][x][y] = 0;
            }

            imgin[i][x][y] = (unsigned char)itemp[i][x][y];
        }
    }
}

/*
printf("¥n<入力 YCbCr 信号(整数値)>¥n");

if(height <= 16 || width <= 16){
```

```
printf("¥n<Y 信号>¥n");  
for(y = 0; y < height; y++){  
    for(x = 0; x < width; x++){  
        printf("%02x ", imgin[0][x][y]);  
    }  
    printf("¥n");  
}  
  
printf("¥n<Cb 信号>¥n");  
for(y = 0; y < height; y++){  
    for(x = 0; x < width; x++){  
        printf("%02x ", imgin[1][x][y]);  
    }  
    printf("¥n");  
}  
  
printf("¥n<Cr 信号>¥n");  
for(y = 0; y < height; y++){  
    for(x = 0; x < width; x++){  
        printf("%02x ", imgin[2][x][y]);  
    }  
    printf("¥n");  
}  
}  
else{  
    printf("画像サイズが大きいため表示しません.¥n");  
}  
*/
```



```
}

void ybr_to_rgb(){
    int i, x, y, j;

    /*
    printf("¥n<入力 YCbCr 信号(整数値)>¥n");

    if(height <= 16 || width <= 16){

        printf("¥n<Y 信号>¥n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
                printf("%02x ", imgin[0][x][y]);
            }
            printf("¥n");
        }

        printf("¥n<Cb 信号>¥n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
                printf("%02x ", imgin[1][x][y]);
            }
            printf("¥n");
        }

        printf("¥n<Cr 信号>¥n");
        for(y = 0; y < height; y++){
            for(x = 0; x < width; x++){
```

```
        printf("%02x ",imgin[2][x][y]);
    }
    printf("¥n");
}
}else{
    printf("画像サイズが大きいため表示しません.¥n");
}
*/

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){
            dtemp[i][x][y] = 0.0;
            for (j = 0; j < 3; j++){
                if (j == 0)
                    dtemp[i][x][y] += ybr_con_rgb[i][j] * (double)imgout[j][x][y];
                else
                    dtemp[i][x][y] += ybr_con_rgb[i][j] * (double)(imgout[j][x][y] -
128);
            }
        }
    }
}

for(y = 0; y < height; y++){
    for(x = 0; x < width; x++){
        for(i = 0; i < 3; i++){

            if(dtemp[i][x][y] > 0.0){
```

```
        itemp[i][x][y] = (int)(dtemp[i][x][y] + 0.5);
    }else{
        itemp[i][x][y] = (int)(dtemp[i][x][y] - 0.5);
    }

    if (itemp[i][x][y] > 255){
        itemp[i][x][y] = 255;
    }else if (itemp[i][x][y] < 0){
        itemp[i][x][y] = 0;
    }

    imgout[i][x][y] = (unsigned char)itemp[i][x][y];
}
}
}

/*
printf("¥n<入力 RGB 信号(整数値)>¥n");

if(height <= 16 || width <= 16){

    printf("¥n<R 信号>¥n");
    for(y = 0; y < height; y++){
        for(x = 0; x < width; x++){
            printf("%02x ", imgout[0][x][y]);
        }
        printf("¥n");
    }
}
```

```
}else{
    printf("画像サイズが大きいため表示しません.\n");
}

*/
```

```
u20216187@gw[33]: ./e x_08_B_01
```

```
header[10]=36 header[11]=00 header[12]=00 header[13]=00
54 バイト
```

<画像の幅>

header[18]=00 header[19]=01 header[20]=00 header[21]=00
256 画素

<画像の高さ>

header[22]=00 header[23]=01 header[24]=00 header[25]=00
256 ライン

<1 画素当たりのビット数>

header[28]=18 header[29]=00
24 ビット

<挿入ビット数>

2 バイト

pepper.bmp をクローズしました。

出力画像の量子化レベル数を入力して下さい(2,4,8,16):3
入力ミス、再入力してください

出力画像の量子化レベル数を入力して下さい(2,4,8,16):16
入力画像データをコピーして出力画像データを作成しました。
出力ファイル名を入力して下さい:pepper_08_B_1.bmp
pepper_08_B_1.bmp をオープンしました。
pepper_08_B_1.bmp をクローズしました。
u20216187@gw[34]: exit

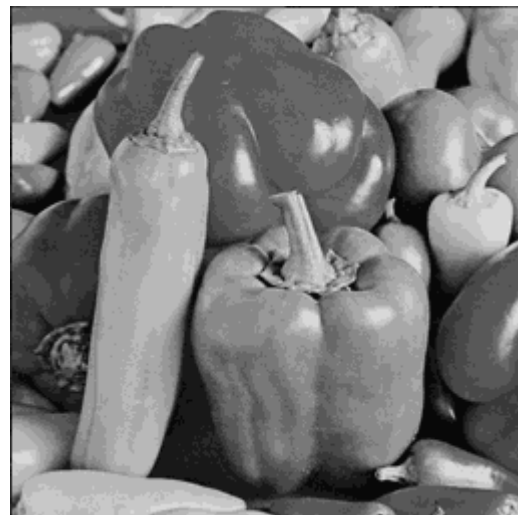
exit

Script done on Thu Dec 1 17:31:46 2022

[添付図]



ここをクリックまたはタップしてテキスト
を入力してください。



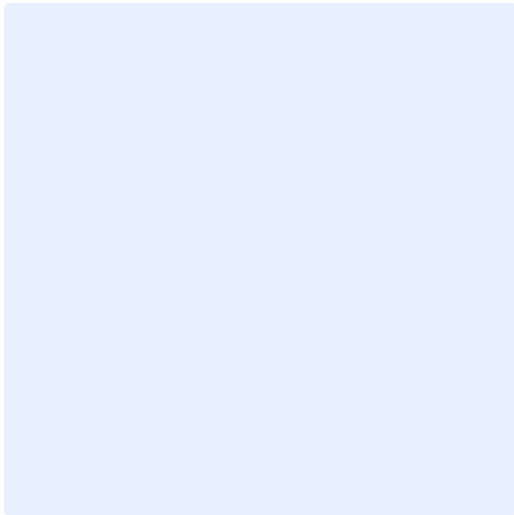
出力画像の量子化レベル数が 16 のとき出
力画像

【問題アイテムを選択してください。-アイテムを選択してください。】

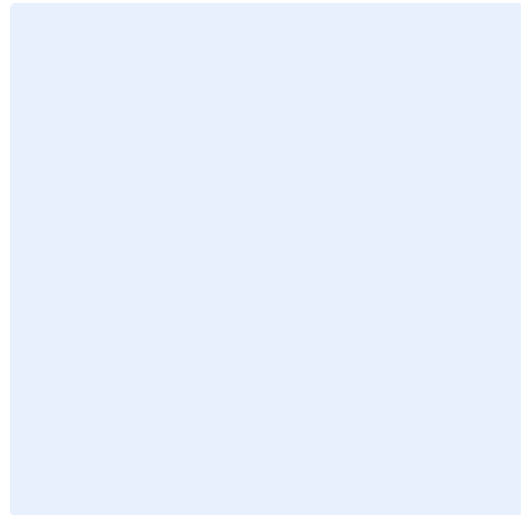
[プログラムリスト・コンパイル結果・実行結果]

ここをクリックまたはタップしてテキストを入力してください。

[添付図]



ここをクリックまたはタップしてテキスト
を入力してください。



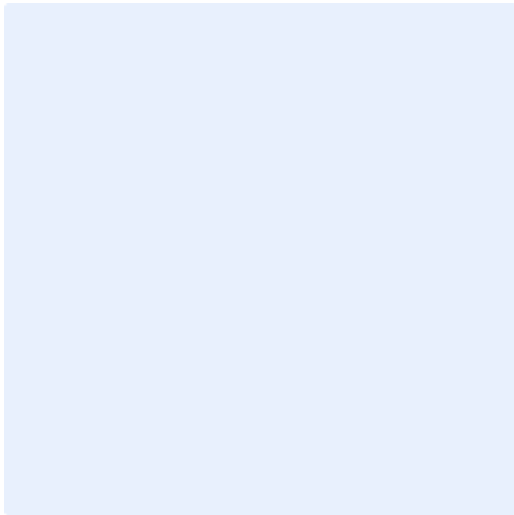
ここをクリックまたはタップしてテキスト
を入力してください。

【問題アイテムを選択してください。-アイテムを選択してください。】

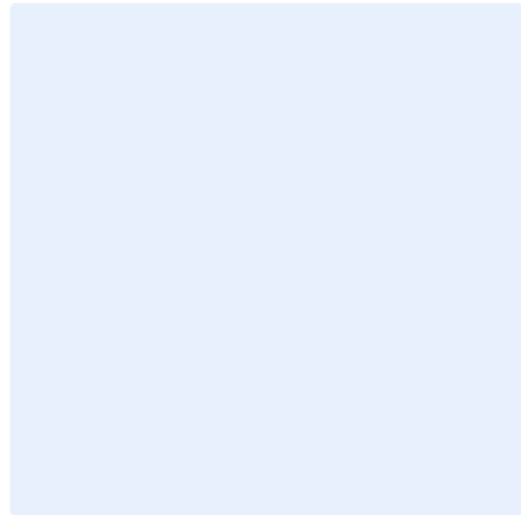
[プログラムリスト・コンパイル結果・実行結果]

ここをクリックまたはタップしてテキストを入力してください。

[添付図]



ここをクリックまたはタップしてテキスト
を入力してください。



ここをクリックまたはタップしてテキスト
を入力してください。