生産専攻 2 年前期　画像情報処理工学

第4回目の目標 ：　ppm 形式ファイルの入出力ができる。 プログラムやデータファイル名 pd

1　前回までの解説

以下のホームページに前回の資料を掲載している。

URL：http://www.ichinoseki.ac.jp/satok/SATOK/ex/index.html

前回の授業のキーワード

インターレース形式のカラー画像保存形式、カラー画像の生成体験

今回は、カラー画像の読み込みと切り取り加工を行う。

2　JPEG 画像を PPM 形式に変換、保存

図１に、本演習で用いる原画像 pd. jpg を示す。ピクセル数は 640、ライン数は 480 である。まず上の URL から画像をダウンロードして、作業ディレクトリに取り込む。

この画像は JPEG 形式で保存されており、このままでは画像演算ができない。いったん ppm 形式に変換する必要がある。

GIMP により、名前を付けて保存の部分で、pd. ppm というファイル名で保存すること。念のため、このファイルを od -c pd. ppm | more でみてもらいたい。先頭に 60 バイト程度（処理系により異なるかもしれない）のヘッダーが見えるはずである。ファイルサイズは、生画像データ 921600 バイトに、そのヘッダー部分が加算された値になる。

図1　原画像 pd.jpg

3　カラー画像を読み込み、そのままの値で保存するプログラム　pd1.c

このプログラムをコピーして一部改良し、改良後は課題番号に従って pd2.c,pd3.c としてファイル保存。

画像ファイルも、この課題のものであることがわかるように pd1.ppm というように、プログラムに対応した名称とする。これらは後で、USB メモリ等でファイル提出を依頼することがある。

画像ファイルからデータを読み込み、加工するために、まず、単純にデータを読み込み、そのままファイル出力するプログラム pd1.c を動かしてみよう。

このプログラムでは、入力ファイルとして fp1、出力ファイルとして fp2 を割り当て、読み込んだカラー画像データはいったん配列 g に格納している。配列のサイズを大きめにとっているが、1000x1000 画素以上の画像データは読み込めない（コアダンプする）ので、サイズのチェックをしている。

また、ppm に変換してファイルサイズを表示すると、何バイトのヘッダーであるか判明するので、そのヘッダー部分を fseek という関数で、読み込みをスキップさせている。

```
* ----------------------------------------
Image Processing pd1.c
File Image Data Processing
Ichinoseki National College Advanced Course
---------------------------------------- */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main(argc,argv)
int argc;
char *argv[];
{
        static int    i,j;
        static int    m;
        static int pixel,line;
        static int g[1000*1000*3]; /* Image data array */
        FILE *fp1,*fp2;         /* fp1:Input fp2:Output Image File */

        /* Command parameter check    */
        if (argc!=5){
                printf("Usage : command input_file pixel line output_file¥n");
                return 1;
        }
        /* Image size parameter load */
        sscanf(argv[2],"%d",&pixel);
        sscanf(argv[3],"%d",&line);

        if (pixel*line >= 3000000){
                printf("Too big image size¥n");
                return 1;
        }

        /* Input image file check */
        if (( fp1=fopen(argv[1],"r") )==NULL){
                printf("Can't open input_file¥n");
                return 1;
        }

        /* Output image file check */
        if (( fp2=fopen(argv[4],"w") )==NULL){
                printf("Can't open output_file¥n");
                return 1;
        }

        /* Skip header data of Input Image */
        fseek(fp1,60,0);
```

```
        /* Load Color Image Data from Input File to Image Array */
        for (j=0; j<line; j++)
        {
        for (i=0; i<pixel; i++)
        {
                g[3*(j*pixel+i)+0]=getc(fp1);   /* Red data load */
                g[3*(j*pixel+i)+1]=getc(fp1);   /*Green data load */
                g[3*(j*pixel+i)+2]=getc(fp1);   /* Blue data load */
        }
        }

        /* ppm header output */
        fprintf(fp2,"P6¥n");
        fprintf(fp2,"# INCT¥n");
        fprintf(fp2,"%s %s¥n",argv[2],argv[3]);
        fprintf(fp2,"255¥n");

        /* Output from Image Array to Image file */
        for (j=0; j<line; j++)
        {
        for (i=0; i<pixel; i++)
        {
                putc(g[3*(j*pixel+i)+0],fp2);   /* Red data output */
                putc(g[3*(j*pixel+i)+1],fp2);   /* Green data output */
                putc(g[3*(j*pixel+i)+2],fp2);   /* Blue data output */
        }
        }

        fclose(fp1);
        fclose(fp2);
        return 0;
}
```

4 画像データを作成し、ファイルを観察してみる

1) pd1.c による画像ファイル pd1.ppm の生成
pd1 pd.ppm 640 480 pd1.ppm

2) データサイズはいくらか？
前回と同様に ls -l コマンドで pd1.pgm ファイルのサイズを確認してみよう。何バイトであったか？

3) 画像データの中身の表示（od コマンド）
od -c pd1.ppm | more で文字による画像データファイルを表示

3) GIMP による表示観察
画像ファイルを「開く」で pd1.ppm ファイルを探し、表示してみよう

5 本日のプログラムの改良課題(必須)

残す時間を、上のサンプルプログラムを改良して、次のプログラム及び画像ファイルを作りなさい。

(1)図2の画像は、図1の原画像の RGB データを変更して出力した画像である。画像サイズは元のままである。
RGB データの対応関係は次の通りである。

入力	出力
Red	Green
Green	Blue
Blue	Red

図2 RGB の割り当て変更を行った画像 pd2.ppm

サンプルプログラム pd1.c の一部を変更して、図2の画像を得るプログラム pd2.c を作成し、画像データ pd2.ppm(640x480 サイズ)を生成しなさい。

(2)図3の画像は図1の原画像の横方向のデータを反転して作成したものである。
図3の画像を得るプログラム pd3.c を作成し、画像データ pd3.ppm を作成しなさい。

図3 横方向反転画像

(3)図4の画像は図1の原画像の座標値(pixel,line)、
(310,90) – (480,360)
の範囲を切り取った画像データ(サイズは 170x270)である。
この画像を得るプログラム pd4.ppm を作成し、
画像データ pd4.ppm を生成しなさい。

図4 画像データの切り取り

この課題は、期末試験の問題のひとつと考えている。
グループで開発してもよいが、どんな原理で画像作成しているか、各自、なぜそうなるか、理由がわかっていることが大事である。

６ プログラムの応用課題

これは自学自習課題である。

(1)画像切り取りプログラムの作成
　上の課題(3)は、プログラム内の値を直接変えて画像を切り取るものである。
また ppm 形式を示すヘッダーも、いつも同じ値であるとは限らない。もっといろいろ使えるようにするために、次のパラメータで、画像切り出しが実行できるように、プログラムを改良しなさい。

コマンド名　原画像ファイル名　pixel line ppm_byte x0 yo xsize ysize　出力ファイル名

ここで、各項目を次のようにする。
　　コマンド名は、コンパイルした実行コマンド(pd5 など)
　　原画像ファイル名は、例えば pd.ppm など
　　pixel は ppm 形式原画像のピクセル数
　　line は ppm 形式原画像のライン数
　　ppm_byte は ppm 形式を示すヘッダーのバイト数
　　x0 は切り出す画像の左上のピクセル座標値
　　y0 は切り出す画像の左上のライン座標値
　　xsize は切り出すピクセルサイズ
　　ysize は切り出すラインサイズ
　　出力ファイル名は出力保存するファイル名

以上が応用課題であるが、余裕があるなら、次の検討をしてみなさい。
　ppm画像形式には、ヘッダーにピクセル数やライン数が書き込まれている。
　また改行コードをたどったり、ファイルサイズを検知できれば、ヘッダーのバイト数がわかる。
　これらのデータを読み込めば上の pixel,line,ppm_byte パラメータを打ち込む必要がなくなる。
　これを実現するにはどうすればよいか、余裕があったら取り組んでみなさい。