生産専攻 2 年前期　画像情報処理工学

第2回目の目標 ：　pgm 形式画像データを作成できる。 プログラムやデータファイル名 pb

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

UNIX、C プログラム、GIMP、pnm 画像形式、グレー画像と罫線

今回も GIMP や UNIX の操作慣れる目的で、単純な幾何図形によるグレー画像作成を行う。

2　パターン図形（pgm 形式）を作成するサンプルプログラム pb1.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 pb2.c,pb3.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 pb1.pgm 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。

以下のものは、図1で示すように4つに区切った左上、右上、左下、右下の領域それぞれを、画素値 255,192,80,0 で塗りつぶした画像である。なお輪郭線は、白地図形がわかるようにしたもので、実際には存在しない。
以下に示すプログラム pb1.c は、図1の画像データを発生させるものである。

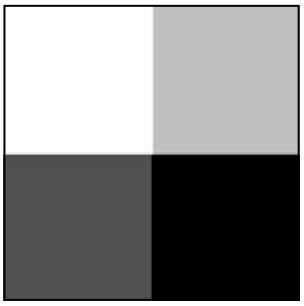

図1　パターン図形 pb1.pgm

```
/* ----------------------------------------
Image Processing pb1.c
Gray Image Pattern Generation
Quad area paint image 255/192/80/0
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,m;
        static int pixel,line;
        FILE *fp1;
```

```
	/* Command parameter check    */
	if (argc!=4){
		printf("Usage : command pixel line output_file¥n");
		return 1;
	}
	/* Image size parameter load */
	sscanf(argv[1],"%d",&pixel);
	sscanf(argv[2],"%d",&line);

	/* Output image file name read */
	if (( fp1=fopen(argv[3],"w") )==NULL){
		printf("Can't open output_file¥n");
		return 1;
	}

	/* ppm header output */
	fprintf(fp1,"P5¥n");
	fprintf(fp1,"# INCT¥n");
	fprintf(fp1,"%s %s¥n",argv[1],argv[2]);
	fprintf(fp1,"255¥n");

	/* Gray image generate */
	for (j=0; j<line; j++)
	{
	for (i=0; i<pixel; i++)
	{
		if (i<=pixel/2 && j<=line/2){
		m=255;
		}
		else if (i>pixel/2 && j<=line/2){
		m=192;
		}
		else if (i<pixel/2 && j>line/2){
		m=80;
		}
		else{
		m=0;
		}
		putc(m,fp1);
	}
	}
	fclose(fp1);
	return 0;
}
```

3 画像データを作成し、ファイルを観察してみる

1) サンプルプログラムによる画像ファイルの生成
pb1 200 200 pb1.pgm
8 ビット、200x200 サイズのグレー画像 pb1.pgm を作成する。生画像データのサイズは 40000 バイト。

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 pb1.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？
40000 を超えたバイト数が、上のリストの/* pnm(pgm) header code output */部分で追加した画像形式を示す目的のヘッダー情報の容量である。

3) 画像データの中身の表示(od コマンド)
od -x pb1.pgm | more で 16 進数の画像データファイルを表示
od -c pb1.pgm | 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 pb1.pgm ファイルを探し、表示してみよう
授業では pgm 形式のまま扱うが、必要に応じて jpeg 画像等に形式変更して保存できる。

4) 画像ファイルサイズを変更し、いろいろな画像を作成してみる
pb1 100 100 pb1.pgm　100x100 サイズの画像になる
pb1 640 400 pb1.pgm　640x400 サイズの画像になる

4 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のように、縦方向に領域を4分割し、4色の画素値 255,192,80,0 で塗りつぶすプログラム pb2.c を作成し、画像データ pb2.pgm を生成しなさい。なお画像サイズは 300 ピクセル x200 ラインである。
なお図2の縁線は説明のために付けたもので、実際には存在しない。
またプログラム内容はサンプルプログラムと同様に、任意の画像サイズに対応し、ピクセルの 4 分の1、2分の1、4分の3、1の各座標値が領域の端の座標であるとする。

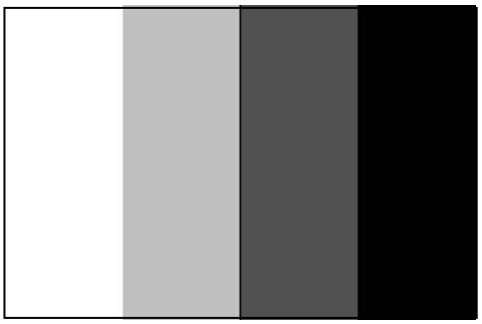

図2　縦方向に4分割したモノクロ画像

(2)図3のように、左端が黒、右端が白の連続階調(グラデーション)となるプログラム pb3.c を作成し、300x200 サイズの画像データ pb3.pgm を作成しなさい。
なお図2の縁線は説明のために付けたもので、実際には存在しない。

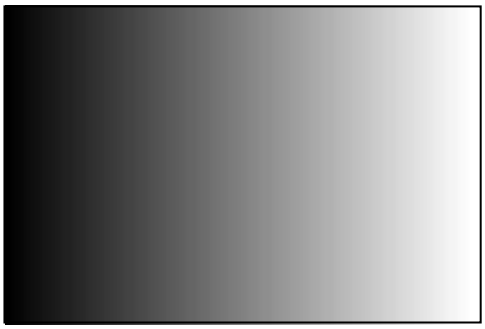

図3　縦方向に4分割したモノクロ画像

5 プログラムの応用課題

これは自学自習課題である。
(1)sin 関数による画像生成 sin(x) (－π≦x≦π)
図4は、ピクセルの中央座標を x=0 とし、左端が－π、右端がπである正弦波(sin 関数)を、最小値を画素値 0、最大値を画素値 255 として作成した 300x200 サイズの画像である。
この画像を発生させるプログラム pb4.c と、画像データ pb4.pgm を作成しなさい。

図4 sin 関数による画像生成

ここで実数変数を使うが、参考までに以下に注意点を示す。
実数変数の宣言例や演算のヒント

```
static double x,pai;
```

内容:x と pai を倍精度実数と定義し、初期値をゼロとする。

```
pai=3.1415926535898;
```

内容:pai の値を 14 桁で設定

```
x=sin((double)(i-pixel/2)*pai/(double)(pixel/2));
```

内容:(double)はキャスト関数で、整数の演算結果を倍精度変数に変換する。
この演算の結果、倍精度変数 x には、－πからπまでの sin の値が入る。
キャスト演算子は、(int)による整数化もある。
なお数学関数を含む C プログラムをコンパイルする場合、

```
cc –o pb4 pb4.c –lm
```

というように、最後に-lm(数学関数ライブラリのリンクの意味)を付けないとコンパイルがうまくいかない場合がある。

(1)cos 関数による画像生成 cos(2y) (－π≦y≦π)

Ppb5.c をさらに改良し、図5のように、縦方向(ライン方向)に cos 関数による画像生成を行うプログラム pb6.c を作成し、画像データ pb6.pgm を生成しなさい。なお画像データは 200 ピクセル×300 ラインとする。

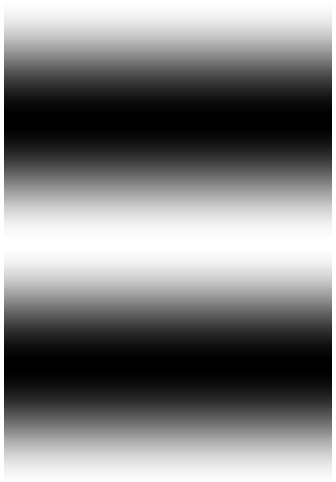

図5 cos 関数による画像生成

この応用課題は、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。
