応用

情報処理演習

（テキスト：第10章）

画像ファイルを扱う

これまでに学んだ条件分岐，繰り返し，
配列，ファイル入出力を使って，画像を
扱うプログラムにチャレンジしてみよう．

画像ファイルからデータ読み込み

- ポータブル・ピックスマップ(PPM)形式は，画像を保存するファイル形式の一つであり，次のように，先頭部分はテキストファイル形式，本体部分はバイナリファイル形式になっているという，2 つの形式が混在したファイルである.

```
P6
#
640 480
255
<画像の本体>
```

ポータブル・ピックスマップ(PPM)形式

- 先頭の1行目は「P6」と書く決まりである.
- 2 行目の部分には,「#」で始まるコメントである. コメントは省略できるし, 複数行あってもよい.
- コメントの次の行は, 画像の横と縦の画素数で, その次の行は階調数である. この場合, 赤と緑と青の各成分が 0 から 255 までの 256 段階の階調であることを示している.

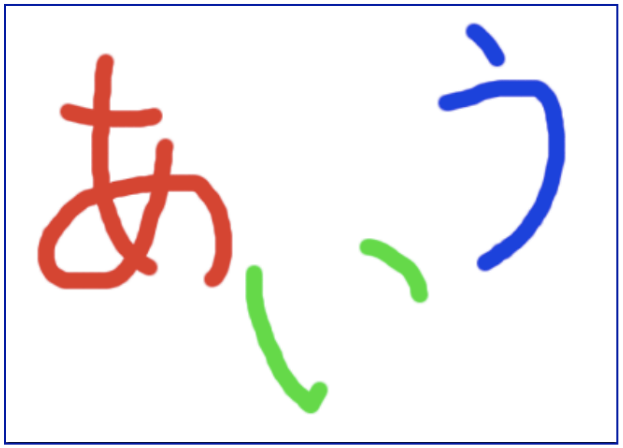
```
P6
#
640 480
255
<画像の本体>
```

特定色の画素の検出

（テキスト134ページ）

- 画像データが保存されているファイルを読んで，特定色の画素の位置を検出するプログラムを作成しなさい．

元画像



赤色ピクセルの検出

生成画像（結果の画像）

元画像で赤色の画素だけ黒にし，それ以外の画素は白にした画像を生成する

特定色の画素の検出

- 画像は平面上に配置された画素の集まり
  - 二次元配列に画像データを入れる.
- 配列の各要素に対して同様の処理を施す場合，繰り返しを使ったプログラムを記述する.
  - 二次元配列を扱うので，二重ループを作成する.
- ある画素の色が特定色であれば，見つけたときの処理，そう出なければ，見つけなかったときの処理を実行するので，条件分岐を使う.
  - ピクセルの値（配列に入っている値）で判断する.

プログラムの流れ

（テキスト136ページ）

1. 定数の定義
2. 各種変数，配列の宣言
3. ファイルから画像データを読み込む
4. 赤色画素の検出，結果の画像の生成（ループ）
5. ファイルへ結果画像の書き出し

画素の値で処理を変える

```
if ((R >= 200) && (G <= 50) && (B <= 50)) {
     「赤い画素」の処理（見つけたときの処理）
} else {
     「赤い画素でない」の処理
}

// ただしR，G，Bは画素の各色の画素値を示す．
```

赤色はRGBでどういう値になるのか?

R >= 200
G <= 50
B <= 50

} このようにしてみる

すべての画素で処理を行う

```
for (i = 0; i < height; i++) {
    for (j = 0; j < width; j++) {
        if ((R >= 200) && (G <= 50) && (B <= 50)) {
            「赤い画素」の処理（見つけたときの処理）
        } else {
            「赤い画素でない」の処理
        }
    }
}
```

画素の値で処理を変える.

すべての画素を対象に値のチェックを行う.

二次元配列 → 二重ループ

定数の定義と変数の宣言

```
#include <stdio.h>

int main (int argc, const char * argv[]) {
    const int MAX_WIDTH = 640;
    const int MAX_HEIGHT = 480;
    int width, height;

    /* 画像を入れる2次元配列の宣言 */
    unsigned char SrcImageR[MAX_HEIGHT][MAX_WIDTH];
    unsigned char SrcImageG[MAX_HEIGHT][MAX_WIDTH];
    unsigned char SrcImageB[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageR[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageG[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageB[MAX_HEIGHT][MAX_WIDTH];

    const char InFileName[] = "ColorChars.ppm"; /* 入力ファイルの名前 */
    const char OutFileName[] = "ExtRed.ppm";    /* 出力ファイルの名前 */
    FILE *f;

    const int LEN = 100;
    char line[LEN];
    int i, j;
```

データの読み込み

```
/* ファイルからの読み込み */
f = fopen(InFileName, "r");
if ( f == NULL ) {
    fprintf(stderr, "ファイル %s のオープンに失敗しました", InFileName);
    return -1;
}
fgets(line, LEN, f);  /* 1行目は読み飛ばす */
fgets(line, LEN, f);
while (line[0] == '#') {  /* コメントは読み飛ばす */
    fgets(line, LEN, f);
}
/* 画像の幅と高さ */
sscanf(line, "%d %d", &width, &height);
fgets(line, LEN, f);  /* データの3行目は読み飛ばす */
for ( i = 0; i < height; i++ ) {
    for ( j = 0; j < width; j++) {
        fread( &(SrcImageR[i][j]), 1, 1, f);
        fread( &(SrcImageG[i][j]), 1, 1, f);
        fread( &(SrcImageB[i][j]), 1, 1, f);
    }
}
fclose(f);
```

条件を満たす画素の検出

```
for (i = 0; i < height; i++) {
    for (j = 0; j < width; j++) {
        if ((SrcImageR[i][j] >= 200) &&
            (SrcImageG[i][j] <= 50) && (SrcImageB[i][j] <= 50)) {
            ExImageR[i][j] = 0;
            ExImageG[i][j] = 0;
            ExImageB[i][j] = 0;
        } else {
            ExImageR[i][j] = 255;
            ExImageG[i][j] = 255;
            ExImageB[i][j] = 255;
        }
    }
}
```

見つけたときの処理
画素を黒色にする.

画素を白色にする.

ファイルへのデータの書き出し

```
    /* ファイルへの出力 */
    f = fopen(OutFileName, "w");
    fprintf(f, "P6¥n");  /* 1行目 */
    fprintf(f, "%d %d¥n", width, height);  /* 2行目 */
    fprintf(f, "255¥n");  /* 3行目 */
    for ( i = 0; i < height; i++ ) {
        for ( j = 0; j < width; j++) {
            fwrite( &(ExImageR[i][j]), 1, 1, f);
            fwrite( &(ExImageG[i][j]), 1, 1, f);
            fwrite( &(ExImageB[i][j]), 1, 1, f);
        }
    }
    fclose(f);

    return 0;
}
```

今日の練習問題

- 今日の練習問題の時間は，これまでに出題されたC言語レポート課題に関するプログラム作成，レポート作成に当ててよい．