第12回
文字列，ファイル入出力，応用

情報処理演習Ⅰ，Ⅴ

（テキスト：第4章，第6章，第9章，第10章）

今日の内容

第3回C言語レポート課題の出題

1. 文字列
2. 文字列の長さ
3. ファイル入出力
4. fgets の振る舞い
5. 応用
6. 今日の練習問題

第3回C言語レポート課題

課題： 今までの総復習
ファイルや画像を扱うプログラムを作成する

形式： プログラムだけを提出するのではなく，プログラムの説明を書くこと．実行結果をキャプチャして貼付けること．

問題： テキスト p.132 の練習問題から1つ以上を選んで答えなさい

〆切： クラスによって案内がある

1. 文字列

（テキスト34ページ）

- 文字列データを扱うときは，文字の配列を使う
  - 配列には，名前とサイズがある
  - 配列を使うために，配列の使用をコンピュータに伝えること（宣言）が必要

宣言

```
char  x[100];
```

文字データ　名前はx　配列のサイズは100

文字列用の配列のサイズ

- 配列の添字は0から(サイズー1)

  例）char x[100]; と宣言したら,

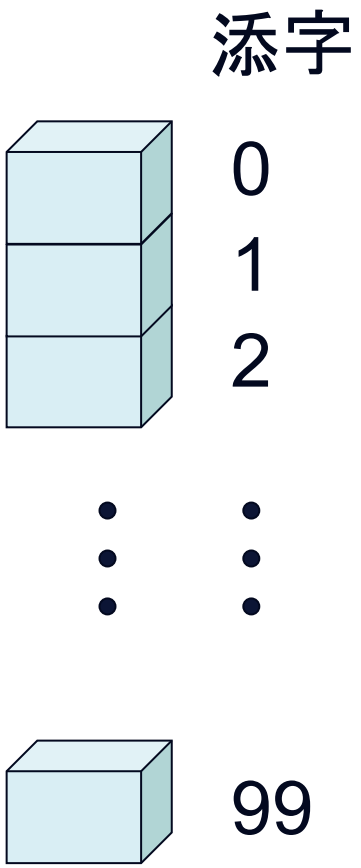
  サイズは 100, 添字は 0 から 99

- 文字列を格納する場所として実際に使えるのは, 0 から(サイズー 2)まで

  例）char x[100]; と宣言したら,

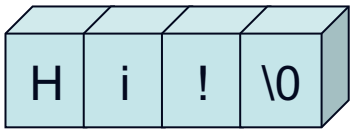
  実際に文字列の格納のために使えるのは, 添字 0 から添字 98 まで（最大 99 文字まで）

  「文字列の末尾」を表す特別な文字定数'\0'(ヌル文字, 値は 0)を入れるのに, 配列要素が1つ使われる



文字列の代入

文字列はダブルクォートで囲む




```
char s[]= "Hi!";
```

初期化

このように書くことはできない

```
char s1[100],s2[100];
strcpy(s1, "Hi!");
```

× `s1= "Hi!";`

```
s2[0]= 'H';
s2[1]= 'I';
s2[2]= '!';
s2[3]= '\0';
```

文字列用のライブラリ関数を使う
（string.hのインクルードが必要）

文字列の終わりを示す文字定数（ヌル(NULL)文字）

文字列の入出力文

- 「書式」と読み込むべき変数名を書く

```
char x[100];
scanf("%s", x);
```

配列名は
ポインタを表す

「&x[0]」と同じ

書式　読み込むべき変数名

配列名の前には「&」を付けない

```
printf("strlen(%s) = %d\n", x, len);
```

書式　表示すべき変数名

2. 文字列の長さ

- 文字列を読み込んで，長さを表示するプログラムを作る

  例）“Computer” の長さは 8

  “プログラミング演習Ｉ” の長さは 20

  - ここでの文字列は，半角の「空白文字」を含まないものとする
  - 半角文字は1，全角文字は2として数える
    - 半角文字は1バイト
    - 全角文字は2バイト

文字列の長さを表示するプログラム

```
#include <stdio.h>
#include <string.h>
int main()
{
    char x[100];
    int len;

    printf("string=");
    scanf("%s", x);
    len = strlen(x);
    printf("strlen(%s) = %d\n", x, len);
    return 0;
}
```

文字列を格納するための配列の宣言
（「char」とあるのは「文字」という意味）

文字列データの読み込み
- 「%s」は文字列の意味
- 配列名「x」は，配列の先頭要素のメモリアドレスの意味

文字列の長さを求める標準関数
（string.h をincludeしておく）

文字列データと長さの表示
- 「%s」は文字列の意味

文字列用のライブラリ関数

（テキスト59ページ）

- 文字列用のライブラリ関数を使うときには，次の1行をプログラムに含めること

```
#include <string.h>
```

- コピー: `strcpy(s, ct)` バッファオーバフローに注意
- 長さ取得: `strlen(cs)`
- 連結: `strcat(s, ct)` バッファオーバフローに注意
- 比較: `strcmp(cs, ct)`
- 検索: `strstr(cs, ct)`

など （cs, ct は const char *, s は char *）

文字列が変わらない

文字列が変わる

例題：文字列の連結

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

int main()
{
    char s1[80], s2[80];

    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    strcat(s1, s2);
    printf("s1=%s, s2=%s\n", s1, s2);
    return 0;
}
```

標準入力から文字列を受け取る

2つの文字列を連結する関数

例題：文字列の比較

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

int main()
{
    char s1[80], s2[80];
    int n;

    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    n = strcmp(s1, s2);
    if (n < 0) printf("%s < %s\n", s1, s2 );
    else if (n == 0) printf("%s == %s\n", s1, s2 );
    else if (n > 0) printf("%s > %s\n", s1, s2 );
    return 0;
}
```

標準入力から文字列を受け取る

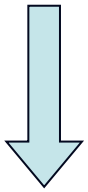
文字列を比較する関数

入力文字列を受け取る時の注意

「文字列の連結」のプログラムに下の部分があるが，これは良いのだろうか？

```
char s1[80];

printf("s1=");
scanf("%s", s1);
```

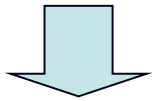


```
char s1[80];

printf("s1=");
fgets(s1, 80, stdin);
```

バッファオーバーフロー

- 入力される文字列が80文字以上の場合は，確保した配列の領域を超えて書き込まれる
- 関数 scanf は s1 の添字の範囲をチェックしない

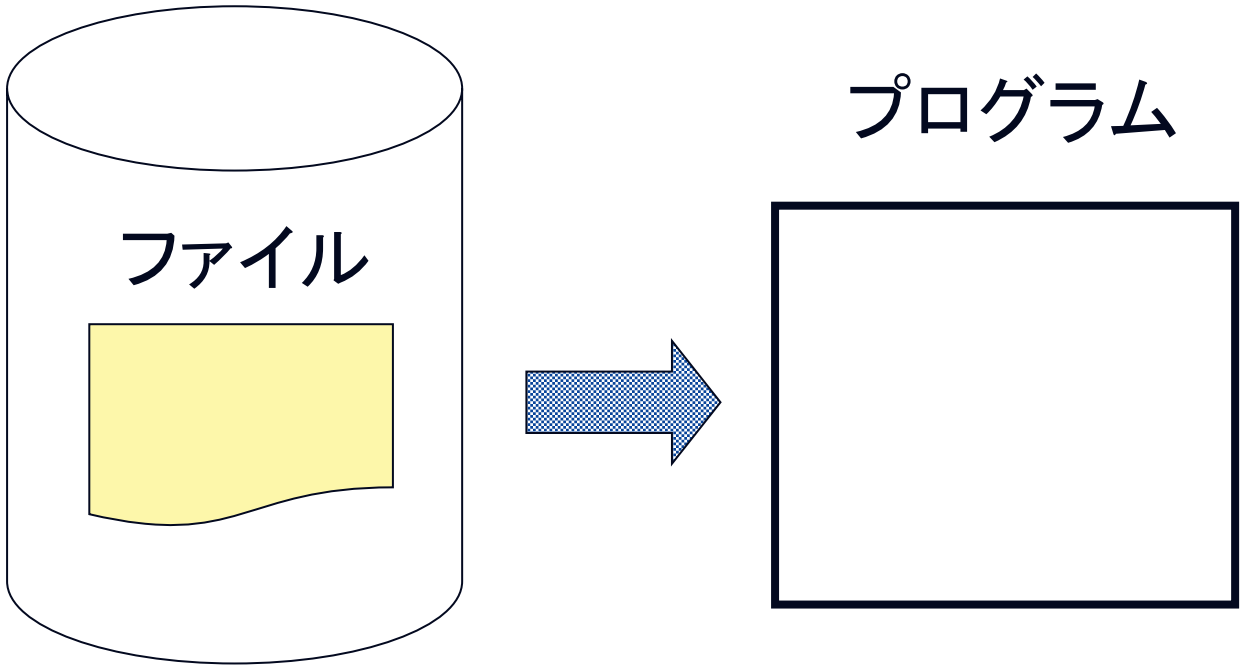


- 入力文字数の上限を指定できる入力関数を使用する

3. ファイル入出力

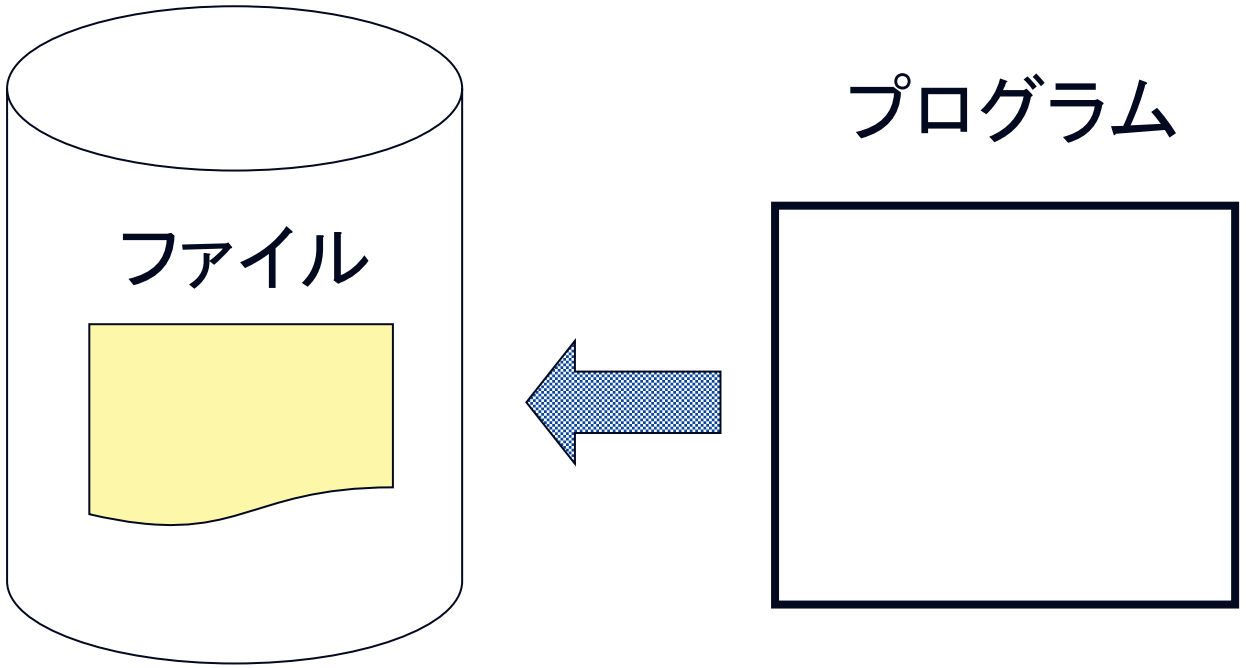
（テキスト121ページ）

ファイルからの読み込み



- ファイルの中身は変わらない

ファイルへの書き出し



- ファイルの中身が変わる
- ファイルは伸び縮みすることがある

ファイルの種類

- テキストファイル
  - 行単位での読み書きに意味がある
  - 人間が「目で見て」読むことができるファイル

```
MPL 40
    pattern1 = 786
    pattern2 = 1
    pattern3 = 979
    pattern4 = 0
```

テキストファイルの例

```
<FF>0<FF><E0>^@^PJFI
F^@^A^B^A^@<96>^@<96
>^@^@<FF><E0>~JFXX^@
^P<FF>00<FF>U^@^C^@^
```

テキストファイルでない
バイナリファイルの例
（画像のファイル）

例題：ファイルからデータ読み込み

- 次のような名簿ファイル（テキストファイル形式）を読み込んで，1列目の氏名と，3列目の住所だけを表示するプログラムを作る
  - 各データは，半角の空白文字（1つまたは複数）で区切られる

```
九大太郎 1200/01/01 福岡市東区箱崎３丁目
情報次郎 1300/12/31 福岡市東区貝塚団地
福岡花子 1800/05/31 福岡市東区香椎浜１丁目
```

３行のテキストファイル

プログラム例

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

データファイルを準備する

（テキストファイル形式）

九大太郎 1200/01/01 福岡市東区箱崎３丁目
情報次郎 1300/12/31 福岡市東区貝塚団地
福岡花子 1800/05/31 福岡市東区香椎浜１丁目

半角の
空白文字

Book1.txt の例
（全角の空白文字が混ざっていると
動かないことがあるので注意）

ファイルから文字列を読み込む

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルオープンに失敗したときのみ実行される部分

whileによる繰り返し部分

NULLの意味

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

int main()
{
    char line[100], nam
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

fopen 関数では「ファイルオープンの失敗」

fgets 関数では「ファイルの終わり」

「==」は等しいという意味
「!=」は等しくないという意味

条件を満たす場合の処理

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

int main()
{
    char line[100], nam
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルオープンに失敗したら，
プログラムが終わる

fopen 関数では「ファイルオープンの失敗」

fgets 関数では「ファイルの終わり」

ファイルの終わりになるまで，
fgets，sscanf，printf を繰り返す

ファイル操作のライブラリ関数

- ファイルのオープンとクローズ
  - fopen　ファイルの読み書きを行う前に，ファイルをオープンしなければならない
  - fclose　ファイルの読み書きが終わったら，ファイルをクローズしなければならない
- ファイルから読み込み
  - fgetc　1文字単位の読み込み
  - fgets　1行単位の読み込み
  - fread　バイト単位の読み込み（1バイト，複数バイト）
- ファイルへ書き出し
  - fputc　1文字単位の書き出し
  - fputs　1行単位の書き出し
  - fwrite　バイト単位の書き出し（1バイト，複数バイト）
  - fprintf　整形した文字列の書き出し

オープンモード

```
in_file = fopen("Book1.txt", "r");
```

ファイル名（文字列）　　オープンモード（文字列）

- ″r″ モード
  - 読み込みモード
  - 引数で指定したファイルが存在しないか，読み込み不可能な場合には，オープンすることができない
- ″w″ モード
  - 書き出しモード
  - 引数で指定したファイルが存在しない場合には，ファイルが新たに作成される
  - ファイルがすでに存在した場合は，ファイル中のデータはすべて捨てられる（ファイルの長さは0になる）

ファイルポインタの使用例

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルポインタ

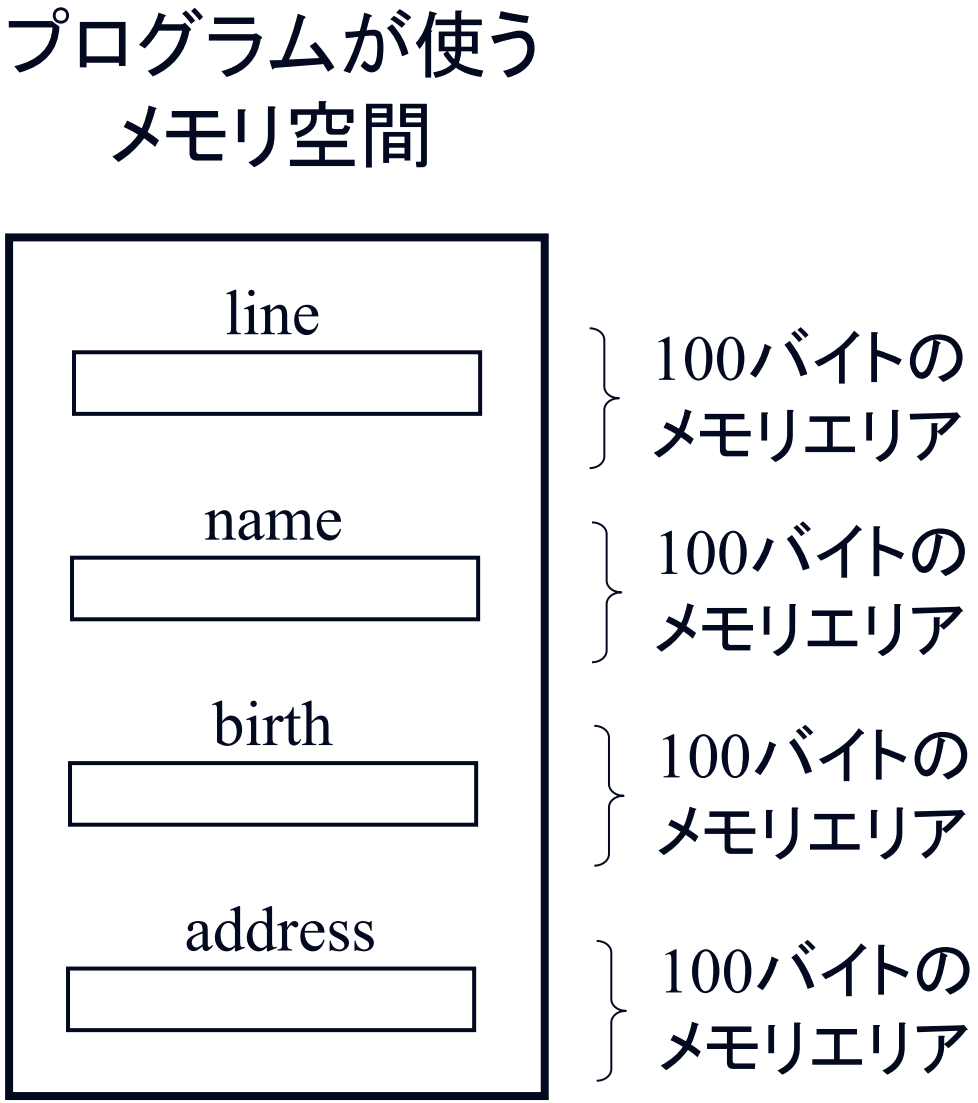
ファイルのオープン

ファイルの読み込み（1行単位）

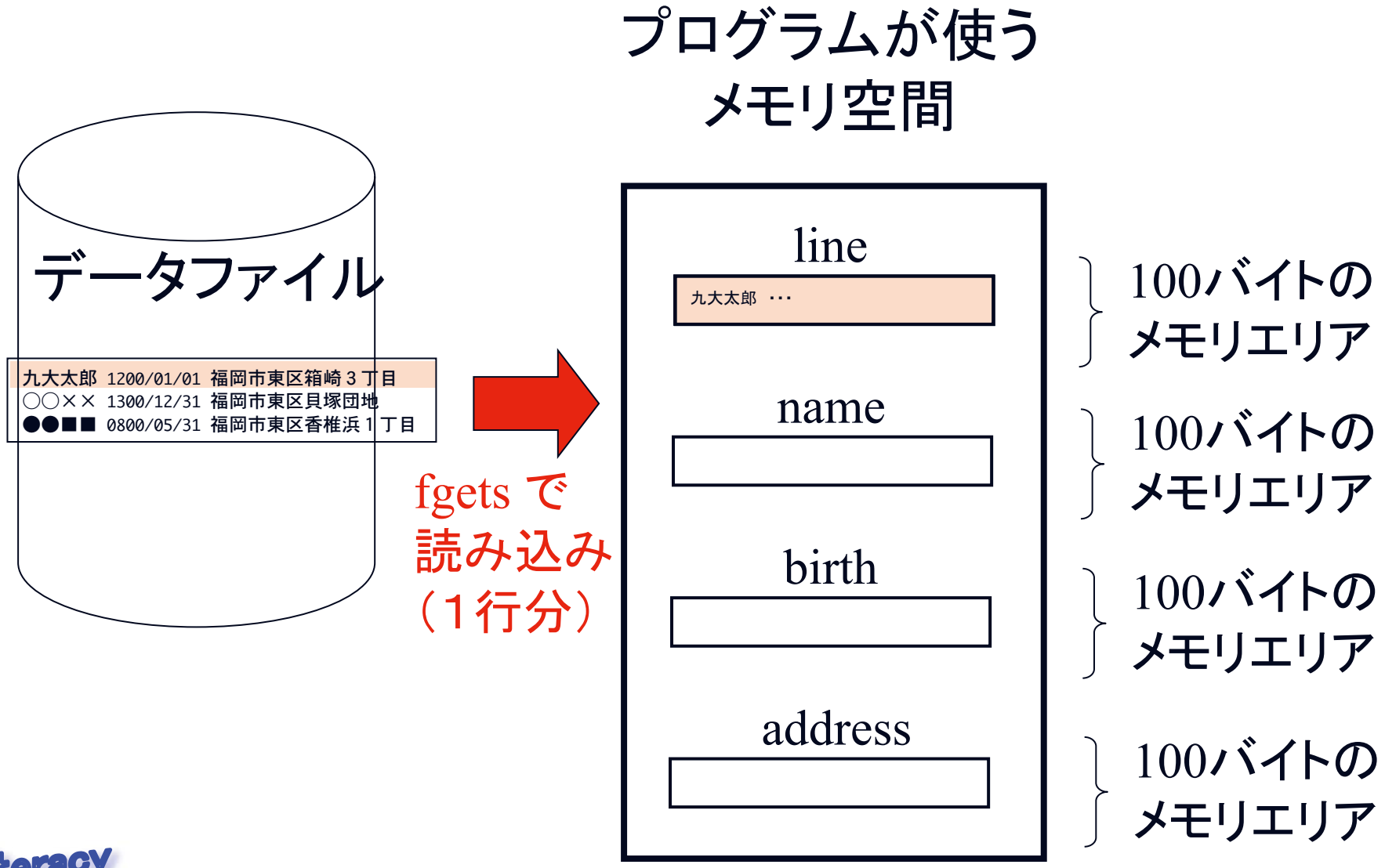
ファイルのクローズ

whileによる繰り返し部分

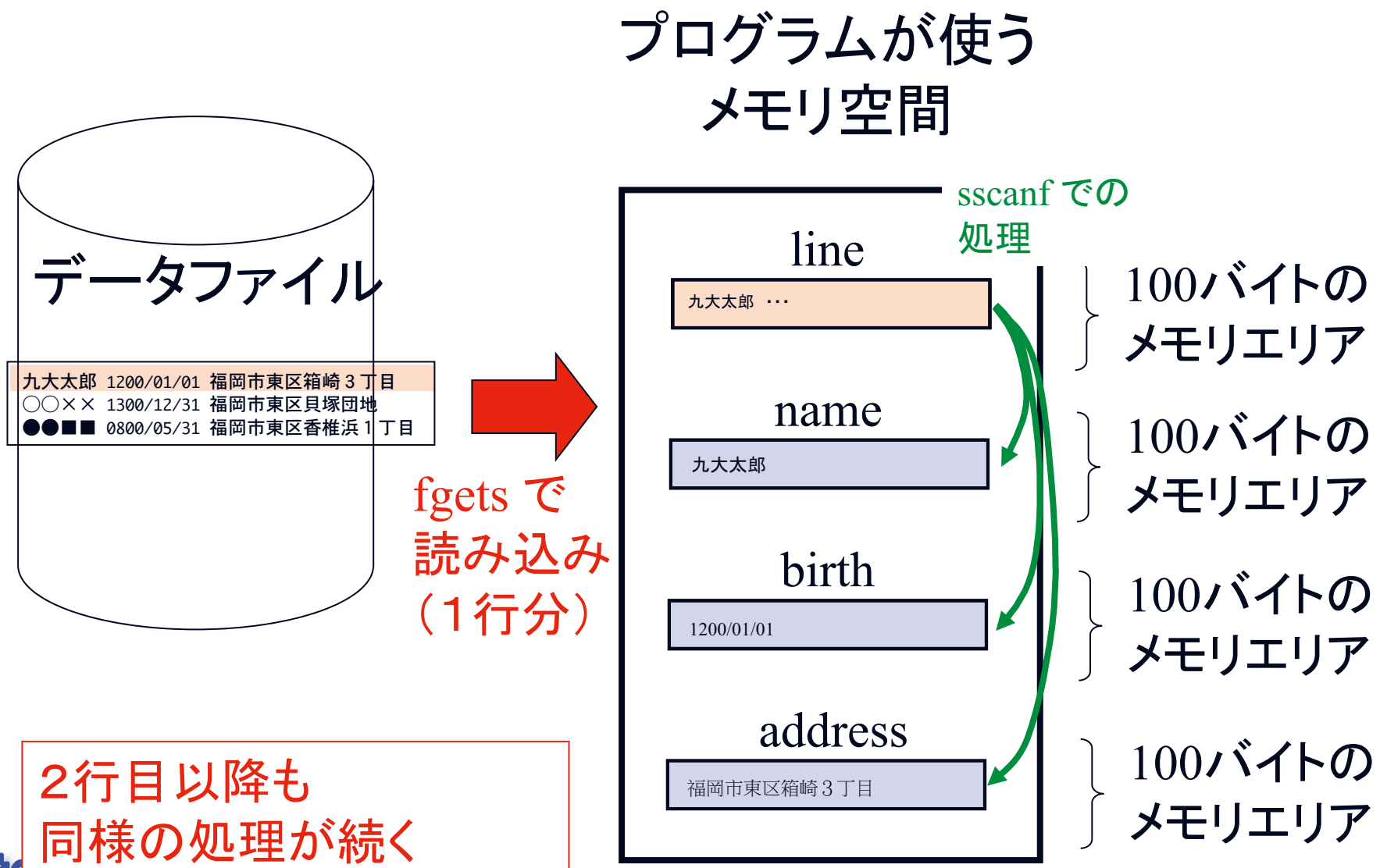
プログラムが行っていること



プログラムが行っていること



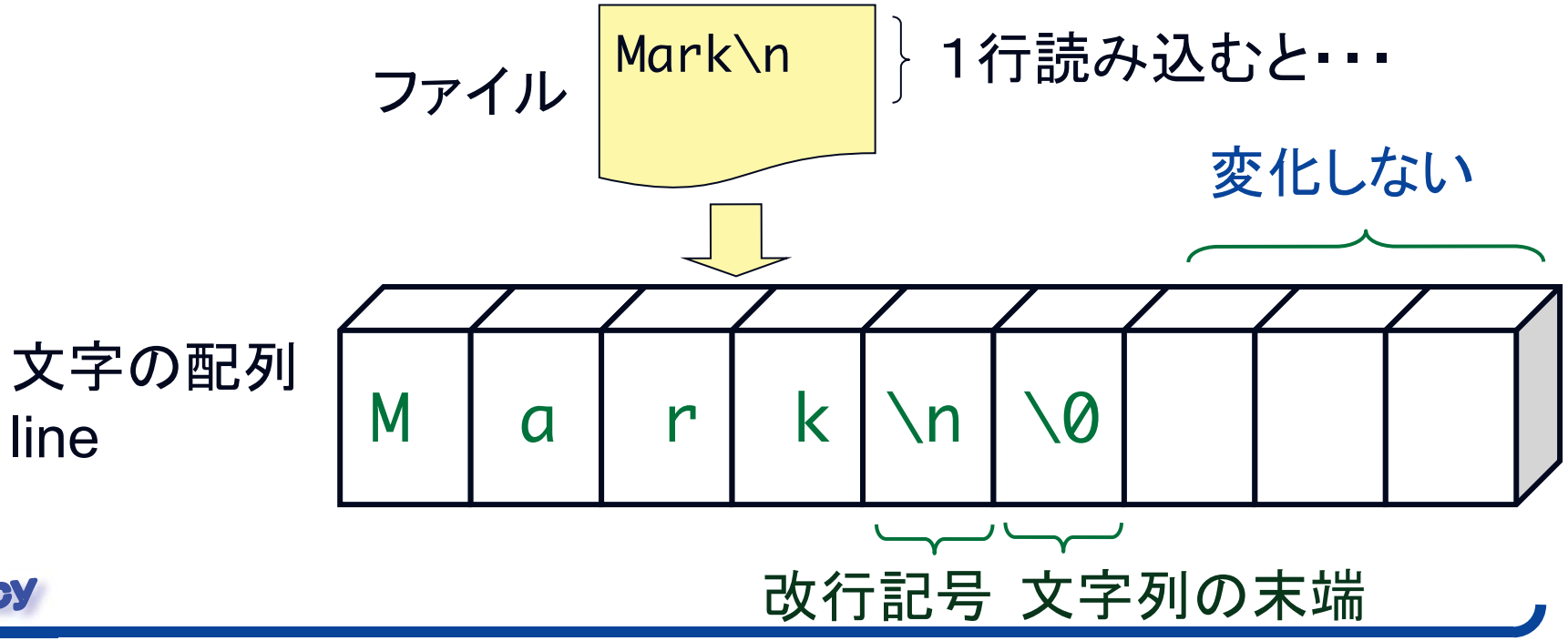
プログラムが行っていること



2行目以降も
同様の処理が続く

4. fgets の振る舞い

- ファイルの1行読み込み
  - ファイルの一行分を読み込んで，末端の¥0を付ける
  - ファイルには，各行の終わりに，改行文字(¥n)が付いている（目には見えない）
  - 読み込み先（文字の配列）のサイズが，ファイルの1行の長さより長いときは，「残りの部分」は変化しない

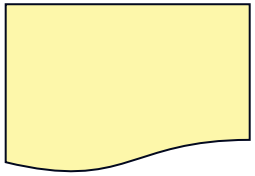


fgets の「100」の意味

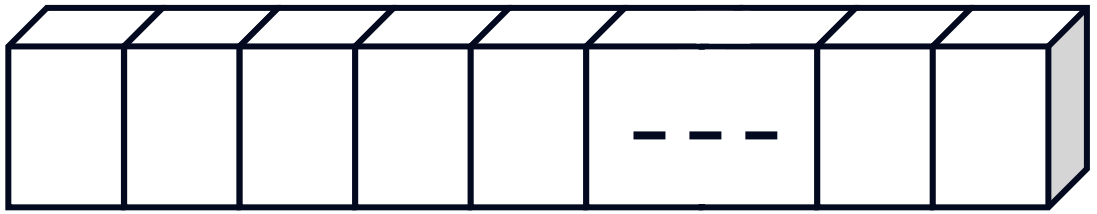
```
fgets( line, 100, in_file )
```

「\0」を含めて100文字に達したら
行末になっていなくても読み込みを終了せよ

ファイル



文字の配列



配列のサイズが100ならば，読み込める文字は99文字まで
（最後に必ず「\0」が付く）

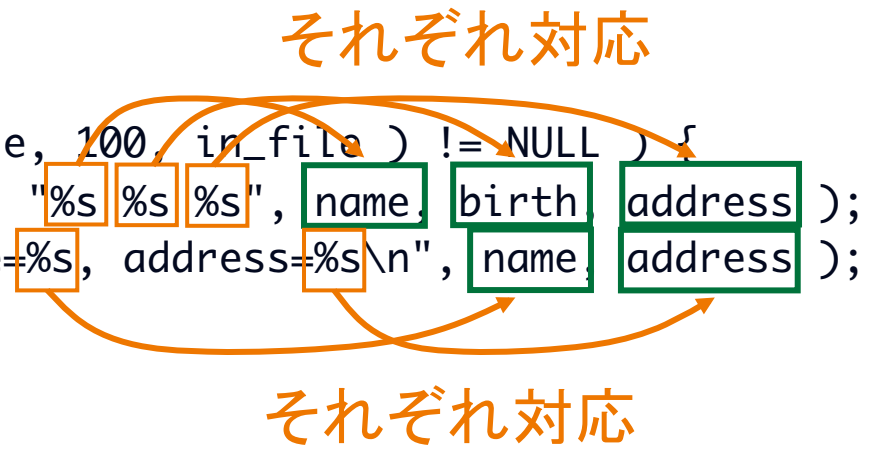
文字列の書式指定文字

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("z:\\Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

char 型の配列については，
sscanf, printf, fprintf
では「%s」を使う決まりになっている



例題：ファイルのコピー

```
#include <stdio.h>
int main()
{
    char line[100];
    FILE *in_file,*out_file;
    in_file = fopen("input.txt","r");
    if ( in_file == NULL ) {
        printf( "fopen_in error" );
        return 0;
    }
    out_file = fopen("output.txt","w");
    if ( out_file == NULL ) {
        printf( "fopen_out error" );
        return 0;
    }
```

ファイルポインタ変数の宣言

入力ファイルのオープン

出力ファイルのオープン

例題：ファイルのコピー（続き）

```
    while( fgets(line,100,in_file) != NULL){
        fputs(line,out_file);
    }
    fclose(in_file);
    fclose(out_file);
    return 0;
}
```

ファイルの終わりに達していないかを調べている

ファイルの
１行読み込み

ファイルのクローズ

ファイルの
１行書き出し

5. 応用
（画像ファイルを扱う）

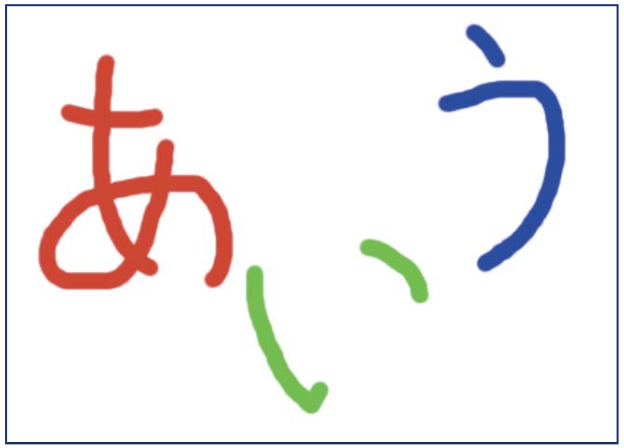
これまでに学んだ条件分岐，繰り返し，
配列，ファイル入出力を使って，画像を
扱うプログラムにチャレンジしてみよう

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み込んで，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;
}
```

6. 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