

cp-14. ファイル処理

(C プログラミング入門)

URL: <https://www.kkaneko.jp/cc/adp/index.html>

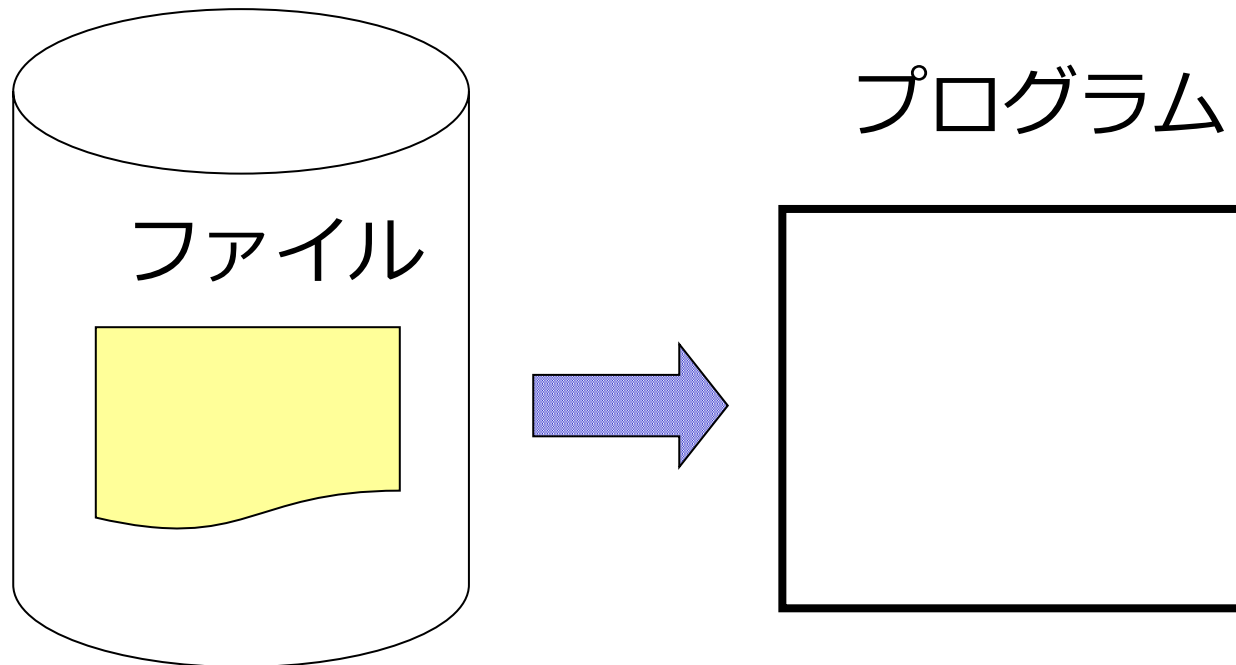
金子邦彦



アウトライン

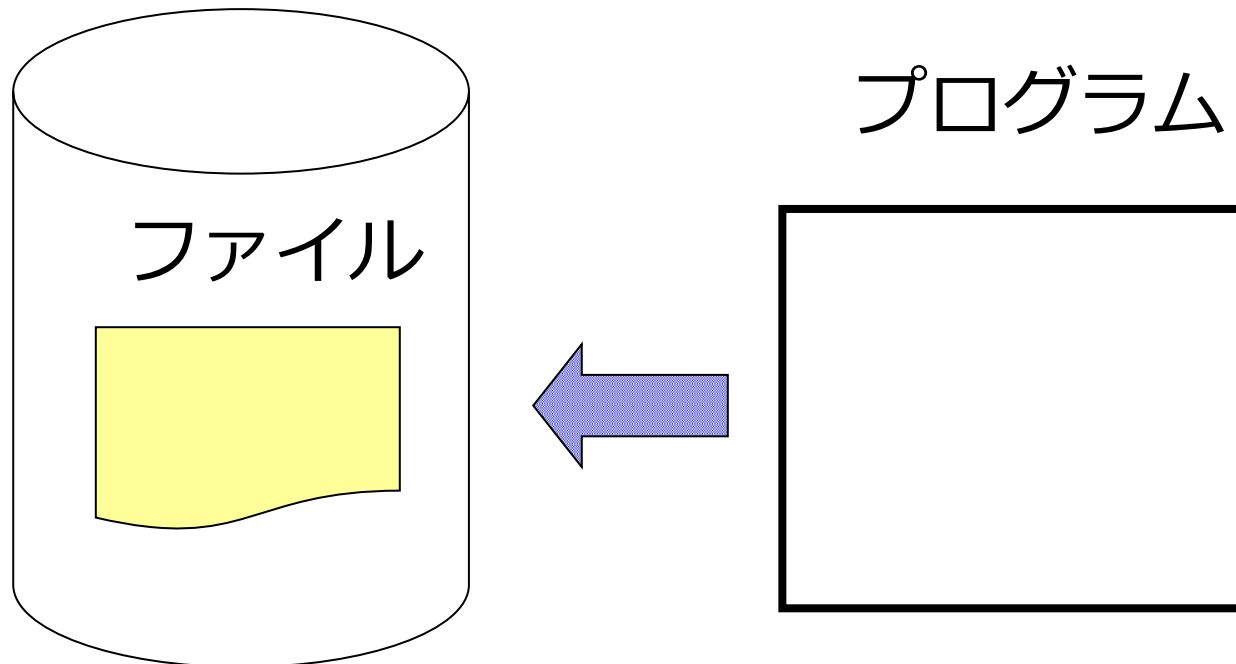
- 例題 1. 1行単位のファイル読み込み
ファイルからの読み込み
ファイルの終わり (EOF ともいう)
- 例題 2. ファイルからのデータ読み込み
ファイル中のデータの取り出し
- 例題 3. 1行単位のファイル書き出し
ファイルへの書き出し
- 例題 4. 3行目を2回読み込む
ファイルのランダムアクセス

ファイル読み込み



- ファイルの中身は書き換わらない

ファイル書き出し



- ファイルの中身が書き換わる
- ファイルは伸び縮みすることがある

- ファイルの読み書きを行うような、簡単なプログラムを書けるようになる
- ファイルを扱う手順（オープン, 読み込み, 書き出し, クローズ）を理解する

例題 1. 1 行単位のファイル読み込み

- ファイルを読み込んで、ファイルの中身に行番号を付けて表示するプログラムを作る。
 - ファイルの読み込みでは、1 行単位の読み込みを行うために `fgets` 関数を使う
 - ファイルのオープンを行うために `fopen` 関数を使い、ファイルのクローズを行うために `fclose` 関数を使う

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

```
#pragma warning(disable:4996)
```

```
int main()
```

```
{
```

```
char line[100];
```

```
int i = 1;
```

```
FILE *in_file;
```

```
in_file = fopen("d:¥¥input.txt", "r");
```

```
if ( in_file == NULL ) {
```

```
printf( "fopen() error" );
```

```
return 0;
```

```
}
```

```
while( fgets( line, 100, in_file ) != NULL ) {
```

```
printf( "[%d]%s", i, line );
```

```
i++;
```

```
}
```

```
fclose(in_file);
```

```
return 0;
```

```
}
```

ファイルポインタ変数の宣言

ファイルのオープン
(ファイルポインタが得られる)

ファイルのオープンに
失敗したかを調べている

ファイルの
1行読み込み

ファイルの終わりに達し
ていないかを調べている

ファイルのクローズ
(ファイルポインタを使ってクローズする)



実行結果例

[1]アレンジメントの特質

[2]・空間を分割

[3]・分割された部分（フェイス）は，凸性を有する（従って有界）か，

[4] 非有界な半空間

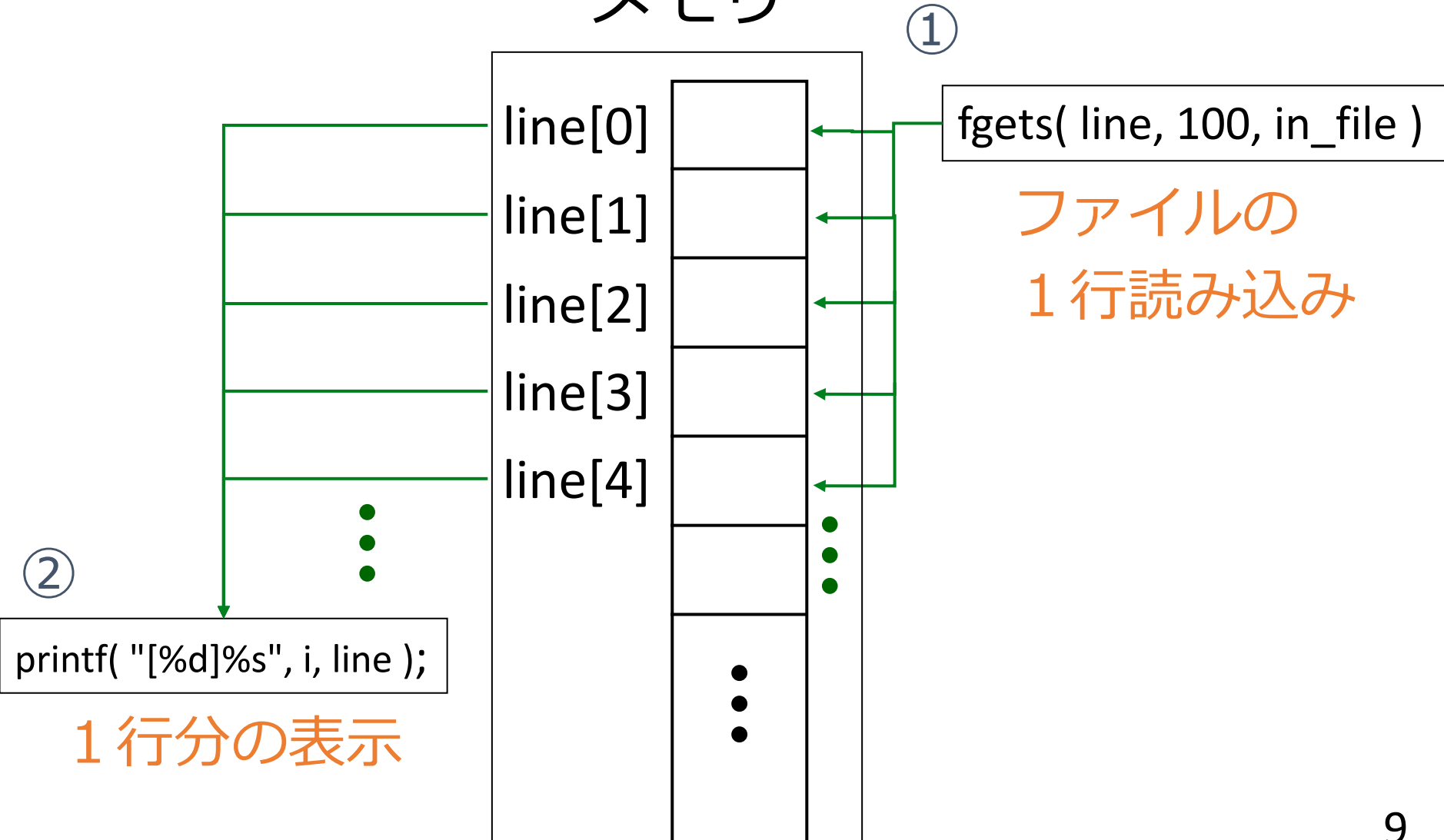
[5]・ボロノイ図を容易に導ける

[6]・単体に容易に分解できる

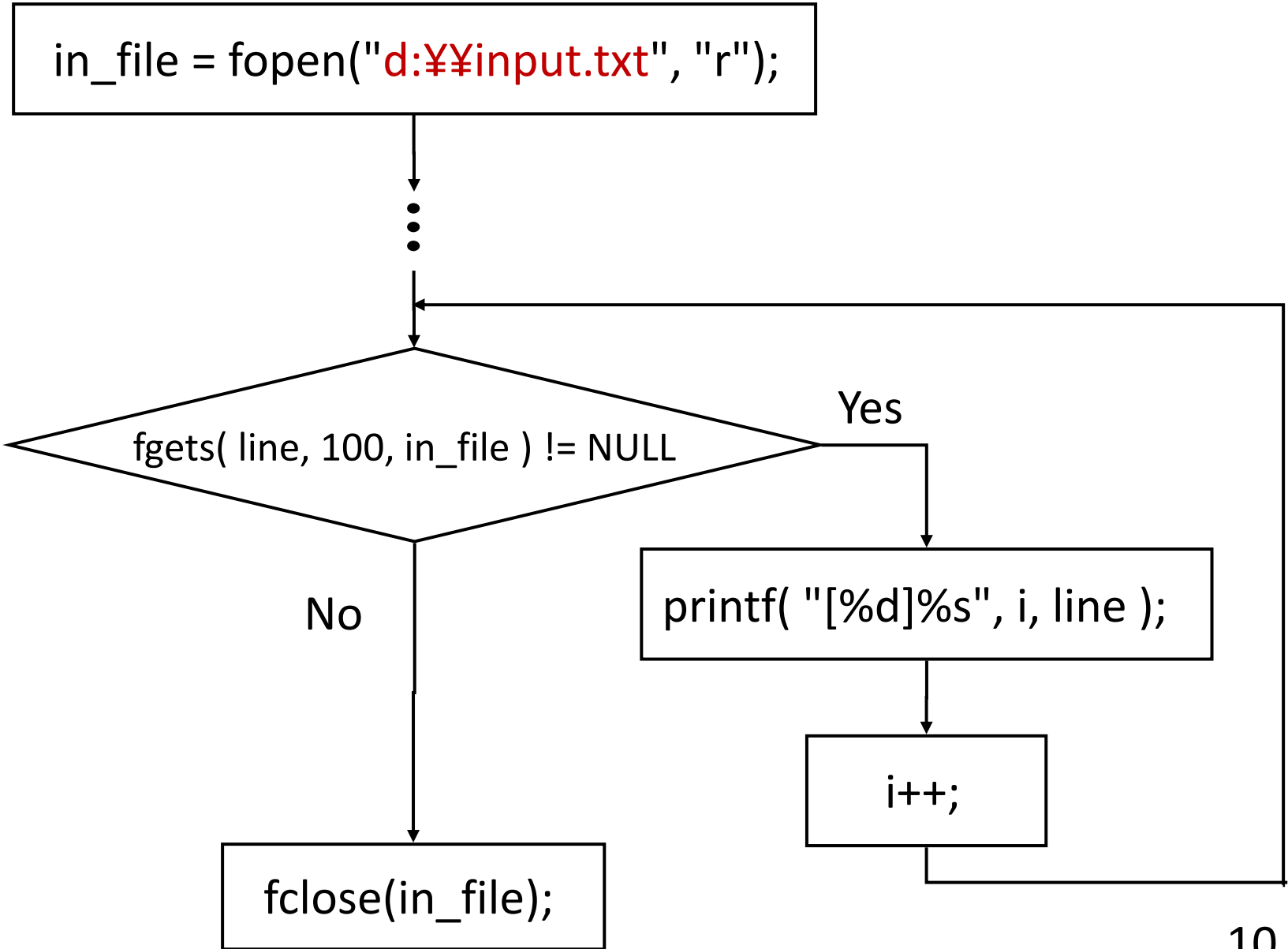
プログラムとデータ



メモリ



プログラム実行順



ファイルのオープンとクローズ

- ファイルのオープン

- ファイルの読み書きを行う前に、ファイルはオープンされねばならない

- ファイルのクローズ

- ファイルの読み書きが終わったら、ファイルはクローズされねばならない

ファイルポインタ

- ファイルポインタ変数の宣言

例) `FILE *in_file;`

「in_file」という名前の付いたファイルポインタ変数を宣言

- ファイルのオープン

例) `in_file = fopen("d:¥¥input.txt", "r");`

↑
オープンすべき
ファイル名

↑
オープンモード
r : 読み込みの意味
w : 書き出しの意味

ファイルのオープンを行うと、ファイルポインタが得られる

- ファイルのクローズ

`fclose(in_file);`

ファイルポインタを使って、ファイルのクローズを行う

ファイルポインタ変数の働き

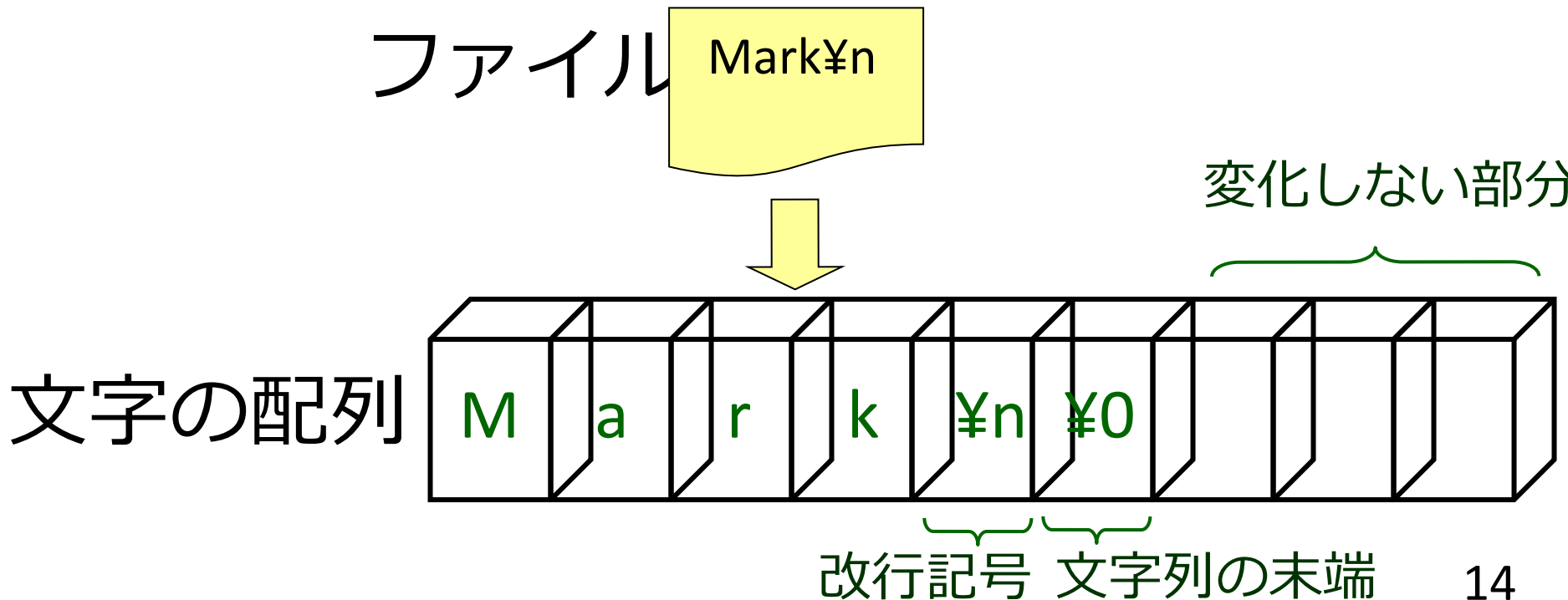


- 現在のファイルのファイル位置指示子
(ファイルの読み書き位置)
- ファイルのオープンモード

など

fgets の意味

- 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、行の終わりに、改行記号 (¥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 で「100」を書く理由



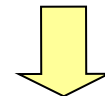
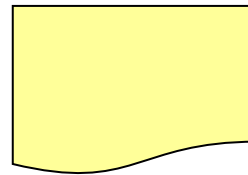
```
fgets( line, 100, in_file )
```

読み込み先の
配列の変数名

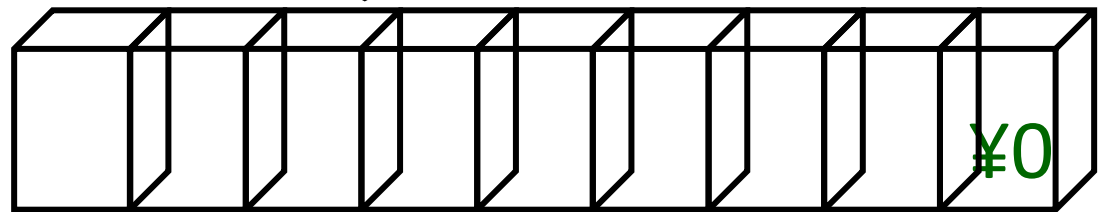
読み込み先の
配列のサイズ

ファイルポインタ
変数名

ファイル



文字の配列



文字列の末端

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

NULLの意味

- fopen 関数では「ファイルオープンの失敗」

例) `fopen("d:¥¥input.txt", "r")`

```
if ( in_file == NULL ) {
```

- fgets 関数では「ファイルの終わり」

例) `fgets(line, 100, in_file) != NULL`

例題 2. ファイルからのデータ読み込み



- 次のような名簿ファイルを読み込んで、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、空白文字で区切られる。
 - 1列目の氏名と、3列目の住所を取り出すために `sscanf` 関数を使う

金子邦彦 1200/01/01 福岡市東区箱崎 092-642-4068

○○×× 1300/12/31 福岡市東区貝塚 092-642-3883

●●■ ■ 800/05/31 福岡市東区香椎 092-642-3884

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
```



```
{
    char line[100];
    char name[100];
    char birth[100];
    char address[100];
    FILE *in_file;
    in_file = fopen("a=d:¥¥Book1.csv", "r");
    if ( in_file == NULL ) {
        printf( "fopen() error" );
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf_s( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s¥n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルポインタ変数の宣言

ファイルのオープン
(ファイルポインタが得られる)

ファイルのオープンに失敗したかを調べている

ファイルの終わりに達していないかを調べている

1行読み込み

1列目, 2列目, 3列目の取り出し

ファイルのクローズ
(ファイルポインタを使ってクローズする)¹⁸

実行結果例

name=金子邦彦, address=福岡市東区箱崎

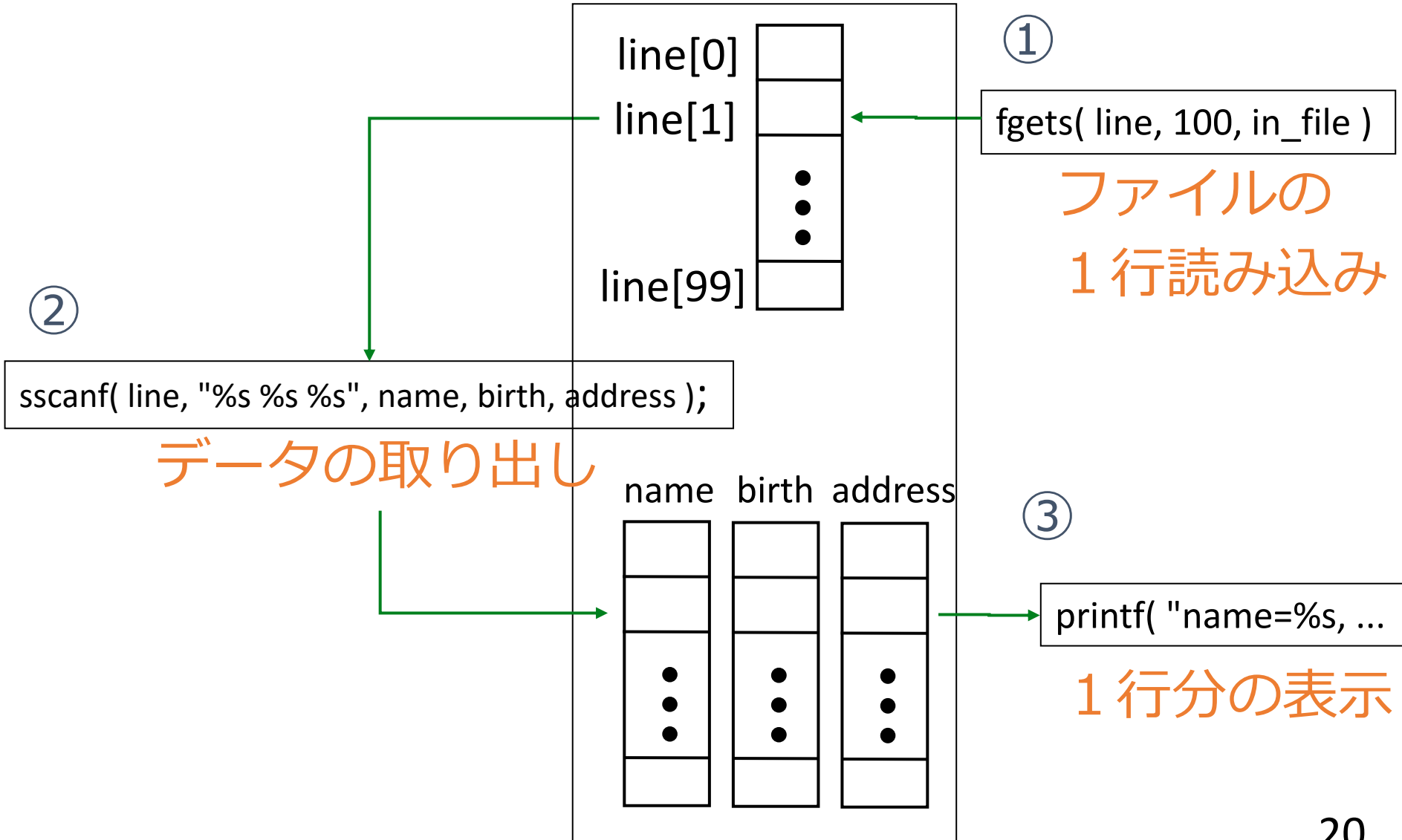
name=○○××, address=福岡市東区貝塚

name=●●■ ■, address=福岡市東区香椎

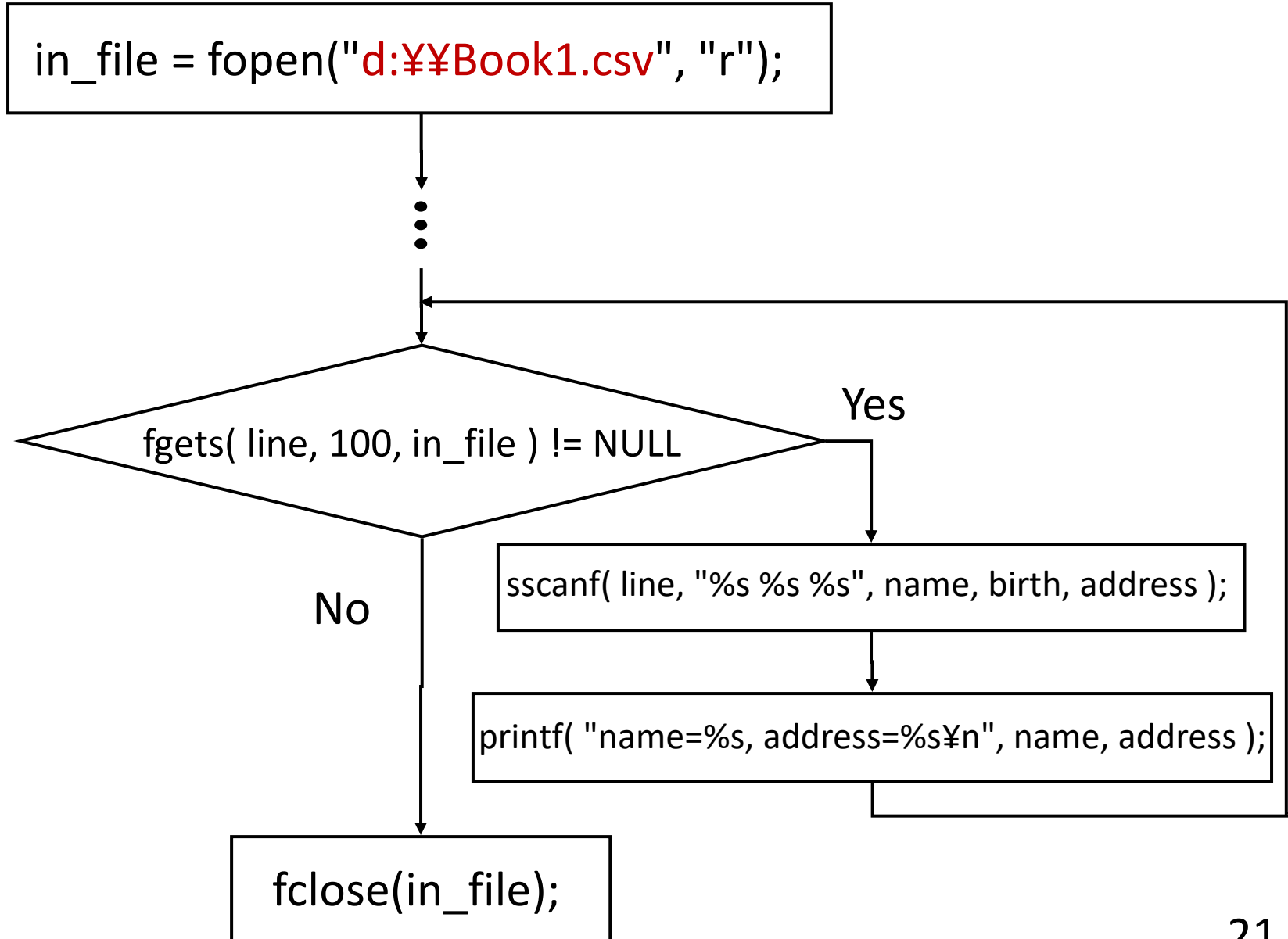
プログラムとデータ



メモリ



プログラム実行順



sscanf 関数

```
sscanf( line, "%s %s %s", name, birth, address );
```

line の中身を調べて、1列目、2列目、
3列目のデータを name, birth, address
に格納する

○○××	1300/12/31	福岡市東区貝塚	092-642-3883	} 1行分
1列目	2列目	3列目	4列目	

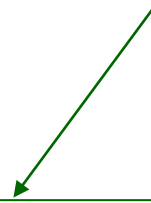
例題 3. 1 行単位のファイル書き出し

- 2つの浮動小数データを読み込んで、sin 関数を使った計算を行い、計算結果をファイルに書き出すプログラムを作る。
 - 「0」と「0.1」を読み込むと、0, 0.1, ... 1.9 について、sin関数を使った計算を繰り返す（繰り返し回数は20回とする）
 - ファイルの書き出すでは、1行単位の書き出しを行うために fprintf 関数を使う
 - 書き出されたファイルは、Microsoft Excel で読み込み可能な形式であること

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

```
struct compute {
    double start_x;
    double step_x;
};
```

start_x と step_x を
キーボードから読み込む関数



```
void input_data( struct compute *data )
{
    printf( "start_x =" );
    scanf( "%lf", &(data->start_x) );
    printf( "step_x =" );
    scanf( "%lf", &(data->step_x) );
}
```


start_x と step_x を使って計算 を行い、結果をファイルに書き込む関数



```
void output_result( struct compute *data, char *file_name )
{
    double x;
    double y;
    int i;
    FILE* fp;
    fp = fopen( file_name, "w" );
    for( i = 0; i < 20; i++ ) {
        x = data->start_x + ( i * data->step_x );
        y = sin( x );
        printf( "x= %lf, y= %lf¥n", x, y );
        fprintf( fp, "x=, %lf, y=, %lf¥n", x, y );
    }
    fclose( fp );
}
```

```
int main()  
{  
    struct compute data;  
  
    input_data( &data );  
    output_result( &data, "z:¥¥data.csv" );  
  
    return 0;  
}
```

関数呼び出しの流れ

main 関数
int main()

関数呼び出し

```
input_data(&data  
);
```

関数呼び出し

```
output_result(&data,"z:¥¥data.csv"  
);
```

input_data 関数

```
void input_data( struct compute *data )
```

戻り

```
return;
```

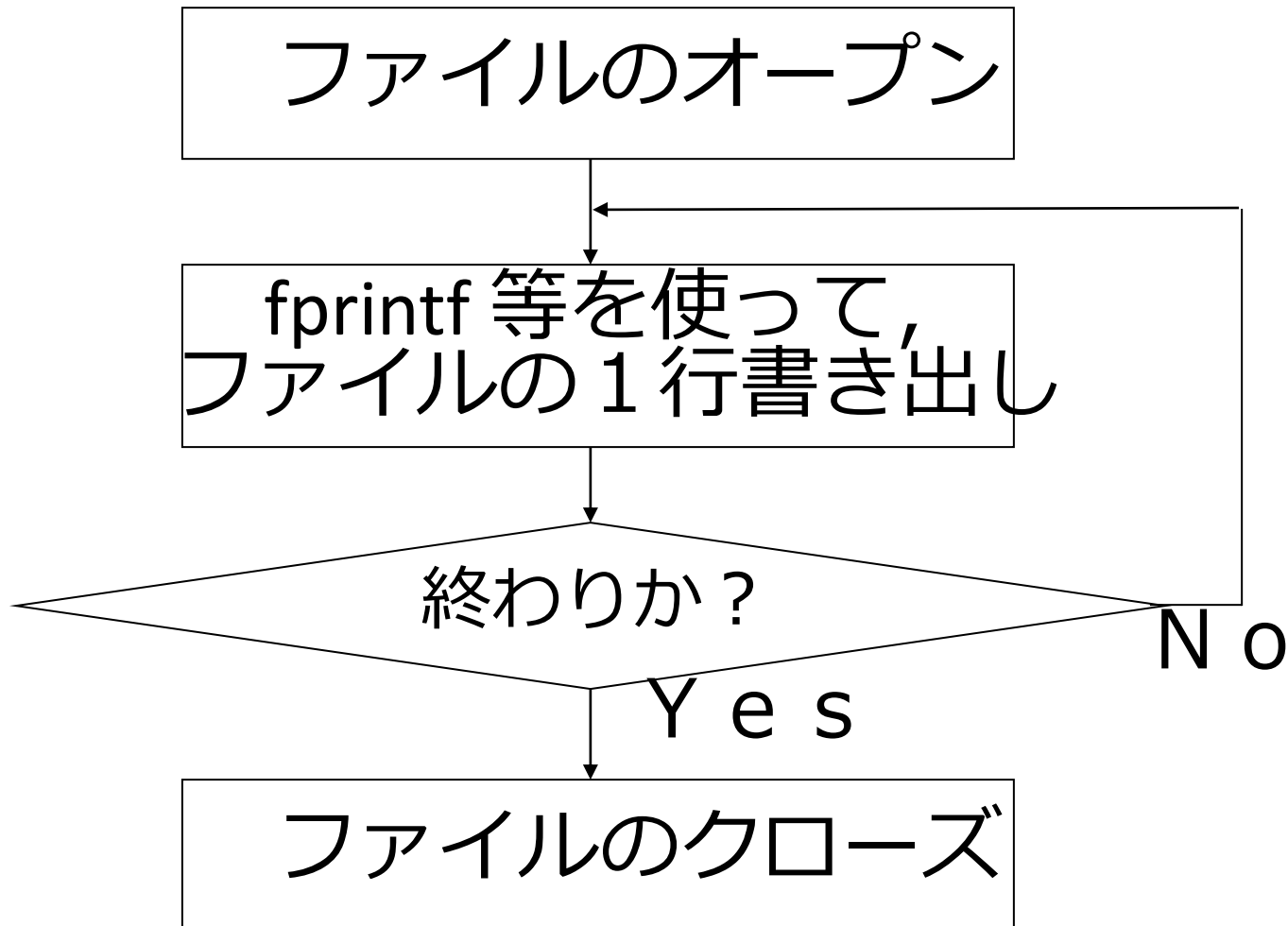
output_result 関数

```
void output_result( struct compute *data, char *file_name )
```

戻り

```
return;
```

1行単位のファイル読み込み の手順



テキストファイルの読み書き



- 読み込み
 - fgets と sscanf を使って, 1行単位で読み込み
- 書き出し
 - fprintf を使って, 1行単位で書き出し

オープンモード

```
fopen( file_name, "w" );
```

ファイル名 オープンモード
(文字列) (文字列)

- “r” モード
 - 読み込みモード
 - 引数file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file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、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）。

課題 3 . ファイルのコピー

- ファイルをコピーするプログラムを作りなさい
 - ファイルを1行単位で読み込んで、別のファイルに1行単位で書き出すことを繰り返す.
 - ファイルの読み込みでは、1行単位の読み込みを行うために `fgets` 関数を使う
 - ファイルの書き出しでは、1行単位での書き出しを行うために `fprintf`関数を使う
 - 2つのファイルを扱うために、2つのファイルポインタ変数の宣言を行うこと
 - 読み込むべきファイルは、Zドライブに事前に作成しておくこと