

# 情報処理実習

## 第11回: ファイル入出力

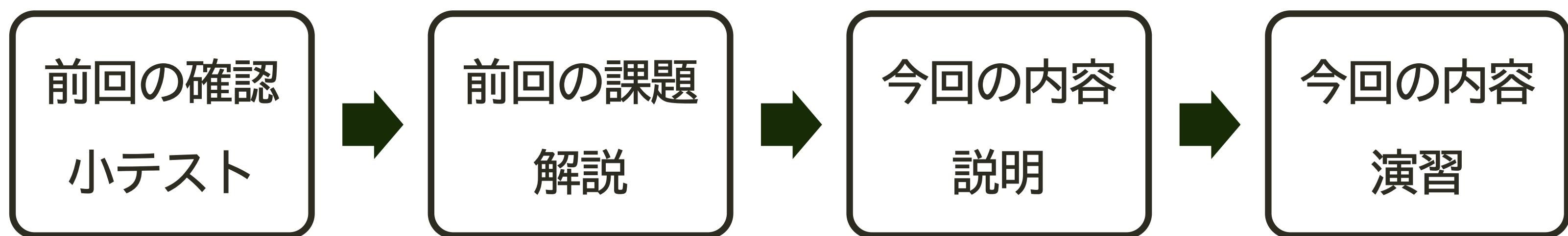
| 担当: 佐藤

2018年12月3日(月)

# アナウンス

- 授業前にやること
  - ・計算機にログオン
  - ・CoursePowerにログイン
- 注意事項
  - ・教室内飲食禁止
  - ・原則として、授業中ではなく休み時間にトイレにいきましょう

# 授業の進め方



# 基本事項の確認小テスト

制限時間: 5分

- ① CoursePowerにログインして、すぐに解答が始められるよう準備する
- ② 開始の合図があるまで、解答を始めずに待機する

注意

「閉じる」ボタンは決して押さないこと！

# ファイルの扱い方



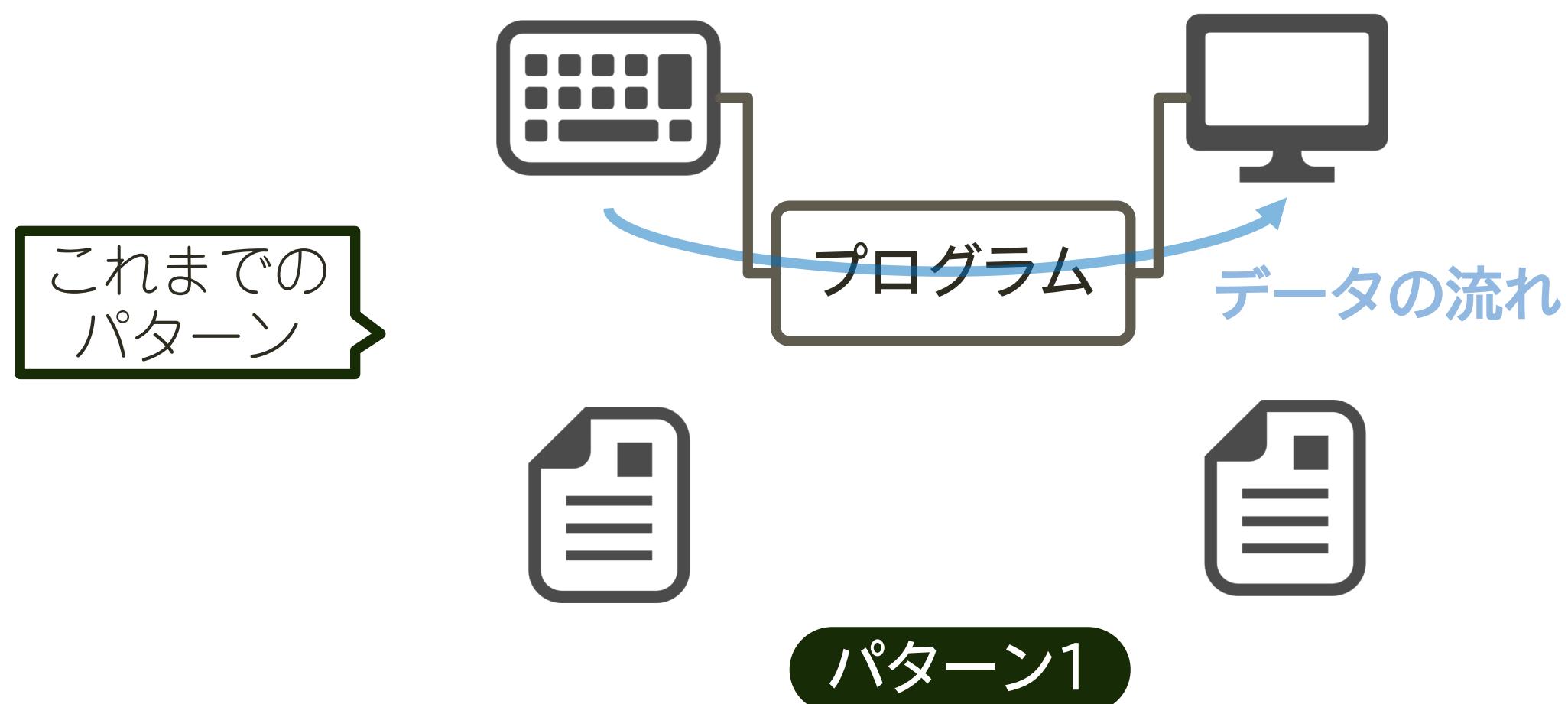
2018年12月3日(月)

# データ処理のパターン

- データの入力元と出力先の組合せとして、次の4パターンが考えられる

パターン	入力	出力
1	キーボード	ディスプレイ
2	キーボード	ファイル
3	ファイル	ディスプレイ
4	ファイル	ファイル

これまでの実習では1のみを扱ってきた。今回は2, 3, 4について学ぶ



# ファイル入出力の手順

- C言語でファイル入出力をする場合、次の手順を踏む必要がある



# ファイルオープン、 ファイルクローズ |

2018年12月3日(月)

# ファイルポインタ

- C言語でファイルを扱う場合、FILE型の変数(ファイルポインタ)を宣言する(※)
  - ・型名(FILE)は、大文字でないとダメ
  - ・変数名の前の\*(アスタリスク)を忘れないこと

## 文法

```
FILE *変数名;
```

※本実習では扱わないが、C言語で高度なプログラムを書くためにポインタの習得は不可欠である

# ファイルオープン

- ファイル入出力では、最初にファイルオープンする必要がある
  - ・ファイルオープンには、`fopen()`を使う
  - ・`fopen()`の第1引数にはファイル名、第2引数にはモードを書く。いずれも“(二重引用符、ダブルクオーテーション)で囲む”
  - ・`fopen()`を使う前にファイルポインタを宣言しておき、`fopen()`の戻り値をファイルポインタに代入する

## 文法

拡張子(例:「.txt」, 「.c」)まで含めて書く必要有り

```
ファイルポインタ = fopen("ファイル名", "モード");
```



プログラム

ファイルオープンによりプログラムとファイルが接続し、データが流れる道(ストリーム)ができる

# モード

- オープンするファイルを読み込む場合と書き込む場合とで、`fopen()`の第2引数に指定するモードが異なる

モード	動作	ファイル有り	ファイル無し
r	読み込み	正常	エラー
w	書き込み	上書き	新規作成

- 既にファイルがある場合、wモードでオープンするとファイルの中身が消えてしまうため注意すること！



# エラー処理

- C言語では、ファイルオープンがうまくいかない場合、自分でエラー処理をする必要がある
- fopen()は、ファイルオープンに失敗するとNULLという値を返す。この値がファイルポインタに代入されていることを利用したエラー処理のお決まりの書き方を以下に示す

## 文法

```
if (ファイルポインタ == NULL) {  
    printf("%sをオープンできません¥n", "ファイル名");  
    return 1;  
}
```

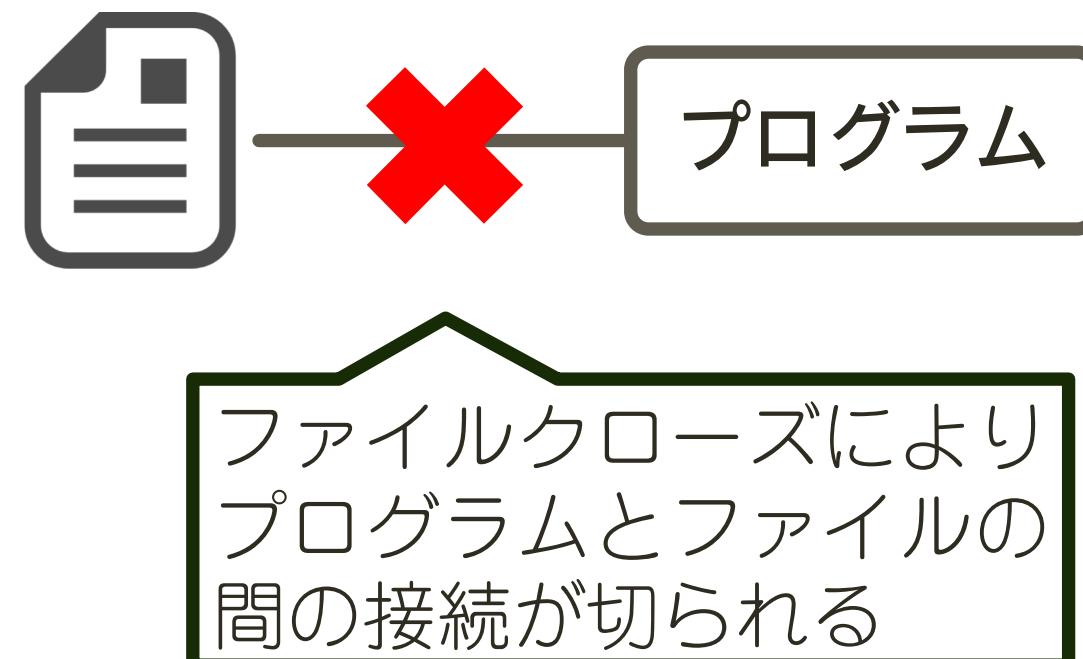
- main()のreturn文の戻り値
  - 0 : 正常終了
  - 0以外: 異常終了

# ファイルクローズ

- ファイル入出力では、最後にファイルクローズする必要がある
  - ・ファイルクローズには、fclose()を使う

## 文法

```
fclose(ファイルポインタ);
```



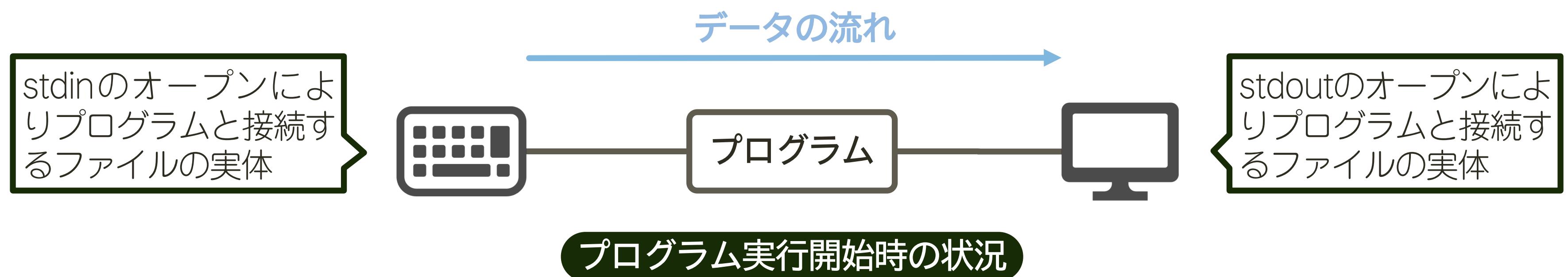
# 標準入力、標準出力



2018年12月3日(月)

# 標準入力, 標準出力

- 標準入力と標準出力という2つのファイルポインタは、プログラム実行開始時に自動的にオープンされ、プログラム実行終了時に自動的にクローズされる
- 標準入力のファイルポインタ名: `stdin`
  - ・オープン→キーボードに接続
- 標準出力のファイルポインタ名: `stdout`
  - ・オープン→ディスプレイに接続



C言語では、電子ファイルだけでなくキーボードやディスプレイなどの機器も「ファイル」として扱われる

# ファイル入出力関数



2018年12月3日(月)

# fscanf()

ファイルからデータを読み込む

- これまで使ってきたscanf()は、キーボードからデータを入力するためのものだった。これに対して、ファイルからデータを読み込むためにはfscanf()を使う
  - fscanf()を繰り返し実行すると、書式に応じてファイルの中身を先頭から順に読み込む

## 文法

```
fscanf(ファイルポインタ, "書式", &変数1, …, &変数n);
```

- 読み込むデータが数値
  - %d(整数), %lf(実数)
- 読み込むデータが文字
  - %c
- 読み込むデータが文字列
  - %s(変数の前の&は不要)

# fscanf()の戻り値

- fscanf()はファイルからデータの読み込みに失敗すると**EOF**を返す
- ファイルの終わりに達すると読み込むデータが存在しないため、読み込みは失敗する→(ファイルの終わりに達するまで戻り値がEOFになることを仮定すれば)戻り値がEOFになるまで繰り返しfscanf()を実行することでファイル中のデータを最初から最後まで読み込むことができる

## 文法

```
while (fscanf(ファイルポインタ, "書式", &変数1, …, &変数n) != EOF) { … }
```

- ・書式の書き方によって、1行ずつ、1…n数値ずつ、1…n文字ずつなど様々な読み込みができる

# fprintf()

ファイルへデータを書き込む

- これまで使ってきたprintf()は、データをディスプレイに出力(表示)していた。一方、データをファイルへ出力する(書き込む)ためには、**fprintf()**を使う

## 文法

```
fprintf(ファイルポインタ, "書式", 変数1, …, 変数n);
```

# fgets()

ファイルから文字列を読み込む

- fgets()は、 gets()と同様に'¥n'に出会うと読み込みをストップする。しかし、 fgets()は、 gets()と違って'¥n'を読み込む
- '¥n'を読み込むと、 '¥n'の後に'¥0'を付加して読み込みを終える

## 文法

```
fgets(文字配列名, 読み込む文字数, ファイルポインタ);
```

読み込む文字数 == nでn-1文字読み込んでも'¥n'が現れない場合、読み込みを終了して末尾に'¥0'を付加する

# fgets()の戻り値

- fgets()はファイルから文字列の読み込みに失敗すると**NULL**を返す
- ファイルの終わりに達すると読み込む文字列が存在しないため、読み込みは失敗する→(ファイルの終わりに達するまで戻り値がNULLになることを仮定すれば)戻り値がNULLになるまで繰り返しfgets()を実行することでファイル中の文字列を先頭行から最終行まで読み込むことができる

## 文法

```
while (fgets(文字配列名, 読み込む文字数, ファイルポインタ) != NULL) { … }
```

# fgets()の使用例(1)

成功

## 例

```
FILE *fp;  
char str1[4], str2[3];
```

'\n' と '\0' の 2 文字分の  
要素数を含めて宣言

...  
ファイルオープンと  
エラー処理をしておく

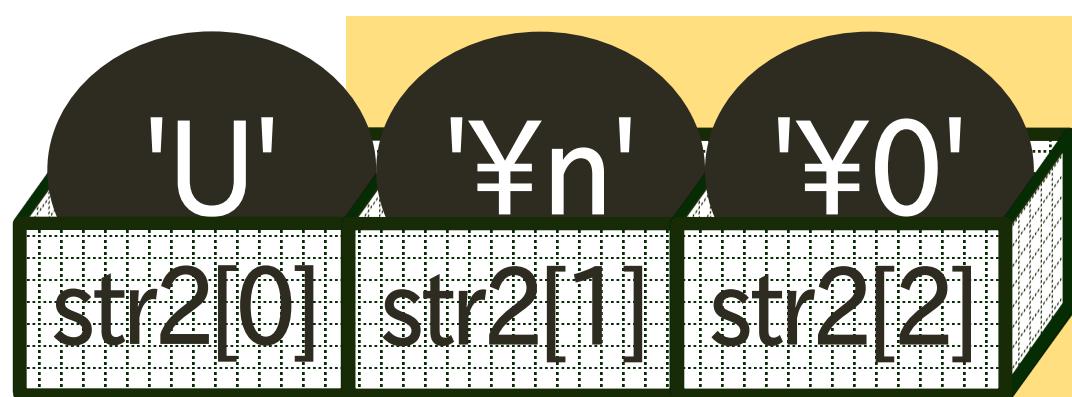
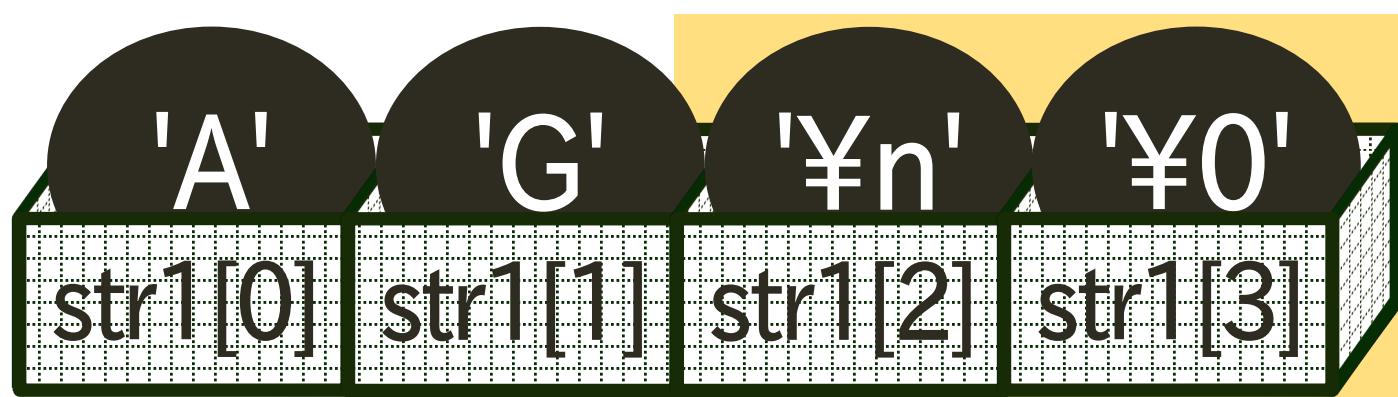
```
fgets(str1, 4, fp);  
fgets(str2, 3, fp);
```

読み込む文字数 → ファイル中の読み込む行の「空白  
文字と目に見える文字の数 + 2」以上の整数を設定

## 読み込むファイル

AG\n  
U\n

各行の最後には見  
えない '\n' がある!



読み込んだ文字を格納する  
文字配列の末尾が「'\n'\0」

# fgets()の使用例(2)

失敗

## 例

```
FILE *fp;  
char str1[4], str2[3];
```

...

間違い!

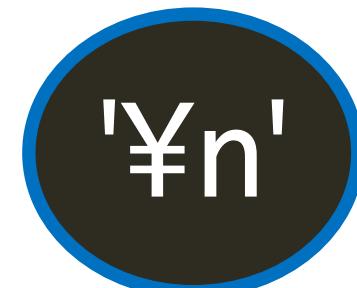
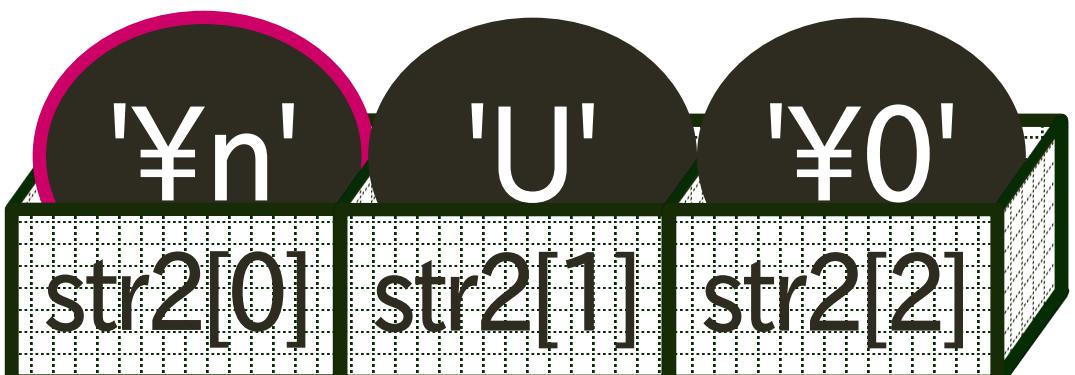
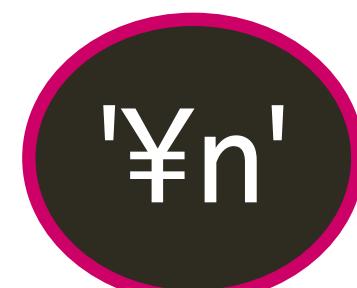
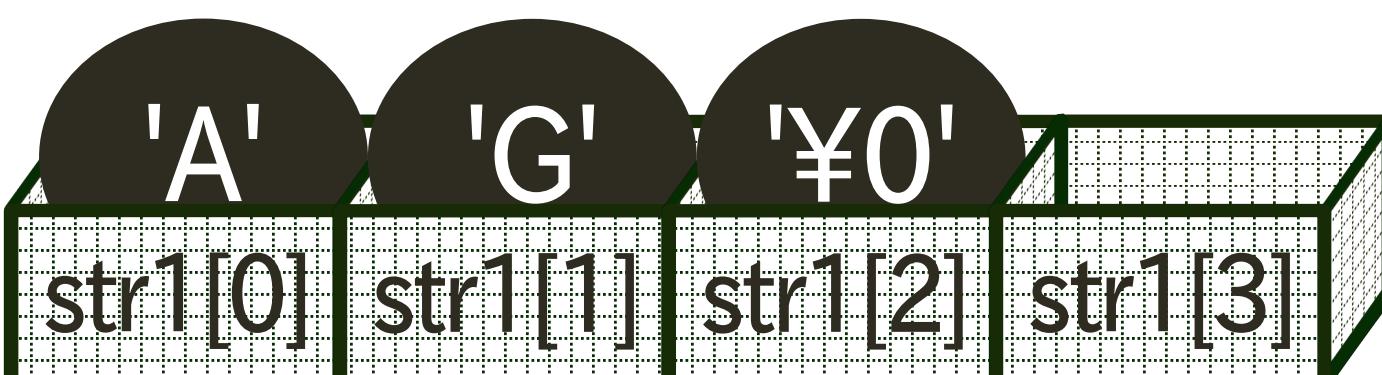
```
fgets(str1, 3, fp);  
fgets(str2, 3, fp);
```

## 読み込むファイル

AG¥n

U¥n

最初のfgets()では読み込まれずに残る。  
次のfgets()で最初に読み込まれる



読み込まれずに  
残ってしまう!

# サンプルプログラム



2018年12月3日(月)

# 読み書きするファイルの場所

- 読み込むファイルは、プロジェクトフォルダの中に入れておく
  - ・「ソリューションエクスプローラー」の中の現在のプロジェクトを選択  
→右クリック→「エクスプローラーでフォルダーを開く」を選択→開かれたフォルダの中に読み込むファイルを入れる
- ファイルオープンで新たに書き込み用のファイルを作成する場合もこの場所(プロジェクトフォルダの中)に作成される
  - ・既にファイルがある場合、wモードでオープンするとファイルの中身が消えてしまうので注意すること!

# サンプルプログラム(1)

キーボードから数値を入力しファイルに出力

## sample11\_1

```
#include <stdio.h>

int main() {
    FILE *fp;
    int v[3];
    char f[] = "data.txt";

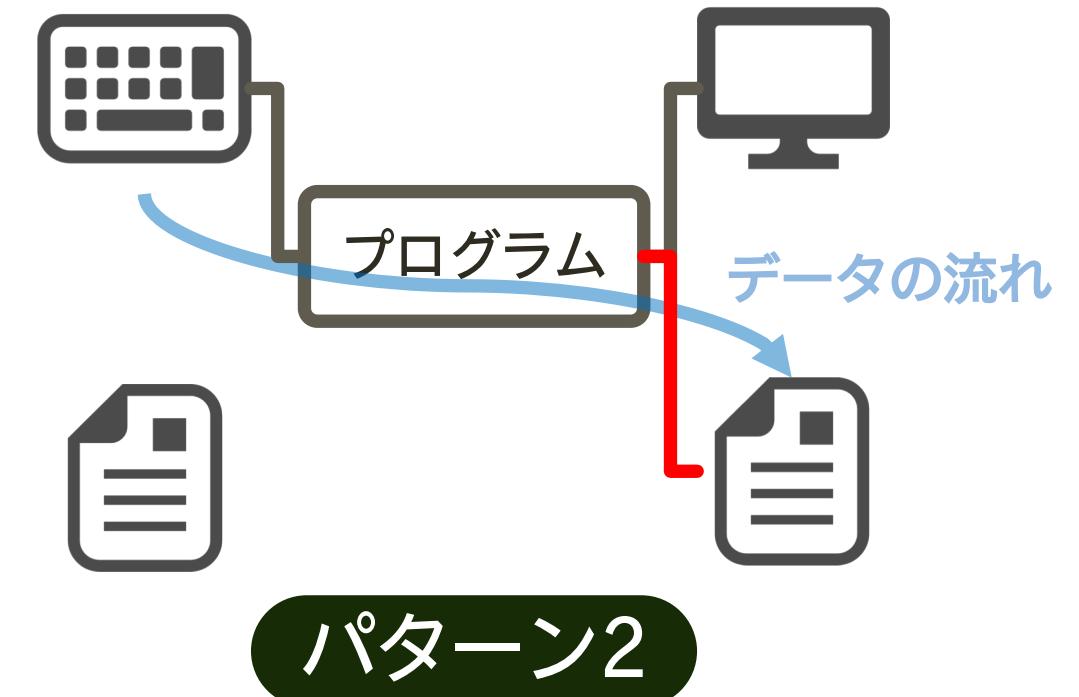
    /* 書込みモードでファイルオープン&エラー処理 */
    fp = fopen(f, "w");
    if (fp == NULL) {
        printf("%sをオープンできません\n", f);
        return 1;
    }  
    「fscanf(stdin, "%d %d %d", &v[0], &v[1], &v[2]);」と同じ

    /* キーボードからvの要素へ数値を読み込む */
    scanf("%d %d %d", &v[0], &v[1], &v[2]);
    /* data.txtにvの要素を書き込み */
    fprintf(fp, "%d %d %d\n", v[0], v[1], v[2]);

    /* ファイルクローズ */
    fclose(fp);

    return 0;
}
```

scanf()はfscanf()のファイルポインタが  
stdinの場合の関数。入力元がキーボードであ  
る場合、わざわざfscanf()を使う必要はない



# 実行結果

- プロジェクトフォルダの中に「data.txt」という名前のファイルが作成される。data.txtの中身を確認すると、「10 20 30」というデータが書き込まれていることが確認できる

## 実行結果(sample11\_1)

10 20 30 ↲

続行するには何かキーを押してください... . . .

**data.txt**

10 20 30

# サンプルプログラム(2)

ファイルから数値を入力しディスプレイに出力

## sample11\_2

```
#include <stdio.h>

int main()
{
    FILE *fp;
    int v[3];
    char f[] = "data.txt";

    /* 読み込みモードでファイルオープン&エラー処理 */
    fp = fopen(f, "r");
    if (fp == NULL) {
        printf("%sをオープンできません\n", f);
        return 1;
    }

    /* data.txtからvに1行読み込み */
    fscanf(fp, "%d %d %d", &v[0], &v[1], &v[2]);
    printf("%d %d %d\n", v[0], v[1], v[2]);

    /* ファイルクローズ */
    fclose(fp);

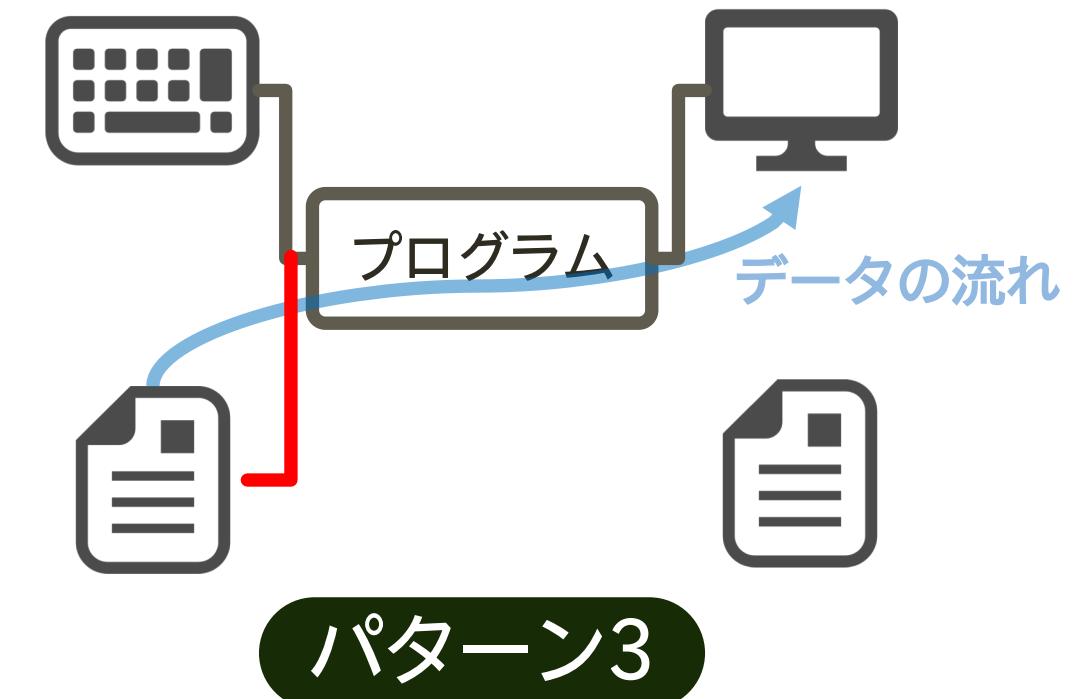
    return 0;
}
```

data.txt

10 20 30

「fprintf(stdout, "%d %d %d", &v[0], &v[1], &v[2]);」と同じ

printf() は fprintf() のファイルポインタが  
stdout の場合の関数。出力先がディスプレイで  
ある場合、わざわざ fprintf() を使う必要はない



# 実行結果

- プログラムを実行する前に、プロジェクトフォルダの中に入らか  
じめ「**data.txt**」を入れておくこと！

## 実行結果(sample11\_2)

```
10 20 30
```

```
続行するには何かキーを押してください... .
```

# サンプルプログラム(3)

ファイルから数値を入力しファイルに出力

## sample11\_3

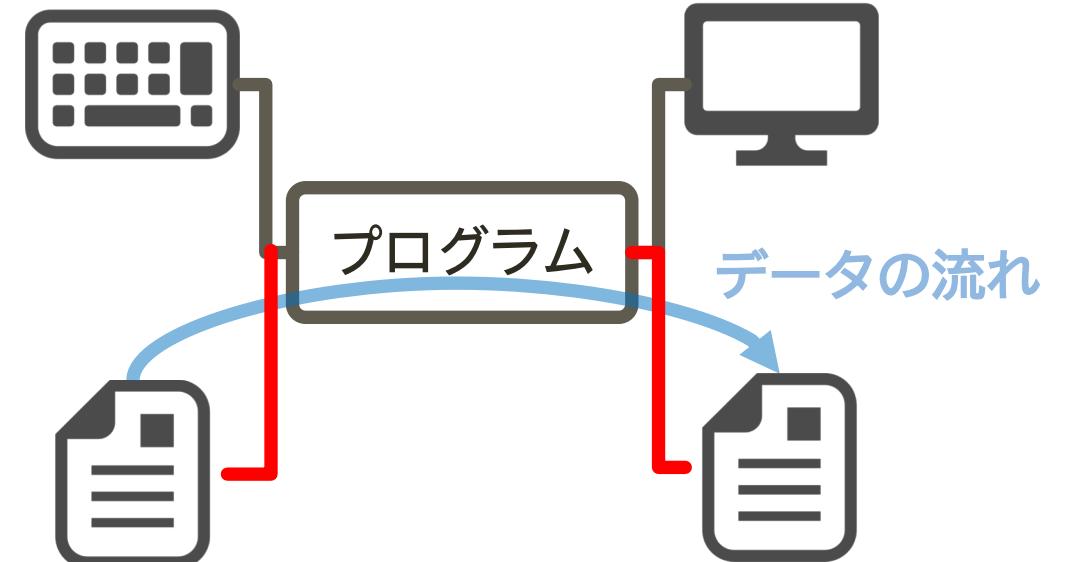
```
#include <stdio.h>
int main()
{
    FILE *fpr, *fpw;
    int v[3];
    char fr[] = "data.txt", fw[] = "copy.txt";
    /* 読込みモードでファイルオープン&エラー処理 */
    fpr = fopen(fr, "r");
    if (fpr == NULL) {
        printf("%sをオープンできません¥n", fr);
        return 1;
    }
    /* 書込みモードでファイルオープン&エラー処理 */
    fpw = fopen(fw, "w");
    if (fpw == NULL) {
        printf("%sをオープンできません¥n", fw);
        return 1;
    }
}
```

data.txt

10 20 30

```
/* data.txtの内容をvへ1行読み込み */
fscanf(fpr, "%d %d %d", &v[0], &v[1], &v[2]);
/* copy.txtにvの各値を書き込み */
fprintf(fpw, "%d %d %d¥n", v[0], v[1], v[2]);
/* ファイルクローズ */
fclose(fpr);
fclose(fpw);

return 0;
}
```



(右へ続く)

# 実行結果

- プログラムを実行する前に、プロジェクトフォルダの中に入らか  
じめ「**data.txt**」を入れておくこと！

実行結果(sample11\_3)

続行するには何かキーを押してください... .

copy.txt

10 20 30

# サンプルプログラム(4)

ファイルから文字列を入力しファイルに出力

## sample11\_4

```
#include <stdio.h>

int main() {
    FILE *fpr, *fpw;
    char str[10];
    char fr[] = "data.txt", fw[] = "copy.txt";

    /* 読込みモードでファイルオープン&エラー処理 */
    fpr = fopen(fr, "r");
    if (fpr == NULL) {
        printf("%sをオープンできません¥n", fr);
        return 1;
    }

    /* 書込みモードでファイルオープン&エラー処理 */
    fpw = fopen(fw, "w");
    if (fpw == NULL) {
        printf("%sをオープンできません¥n", fw);
        return 1;
    }
}
```

data.txt

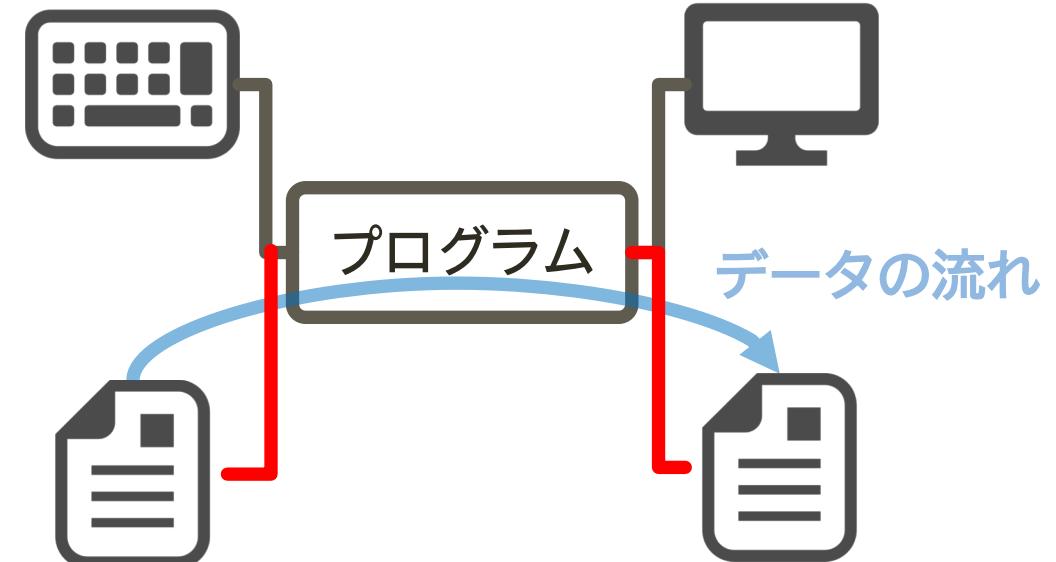
10 20 30

```
/* data.txtの内容をsへ1行読み込み */
fgets(str, 10, fpr);
/* copy.txtにstrの中身を書き込み */
fprintf(fpw, "%s", str);

/* ファイルクローズ */
fclose(fpr);
fclose(fpw);

return 0;
}
```

'¥n'不要



(右へ続く)

※ファイル中の読み込む行の「空白文字と目に見える文字の数+2」以上の整数を設定

# 実行結果

- プログラムを実行する前に、プロジェクトフォルダの中に入らかじめ「**data.txt**」を入れておくこと！

## 実行結果(sample11\_4)

続行するには何かキーを押してください... .

copy.txt

10 20 30

sample11\_3は、ファイルの内容を「数値」として読み書きした。これに対して、sample11\_4は、ファイルの内容を「文字列」として読み書きする。同じパターンであり結果もまったく同じだが、入出力データの種類が異なる

# 課題



2018年12月3日(月)

# レポートの作成

① レポートの冒頭に以下を適切なレイアウトで書く

- 「情報処理実習第11回課題レポート」というタイトル
- 学生番号
- 氏名

② 課題ごとに以下を載せる

- 作成したプログラムのソースコード
- 作成したプログラムの実行結果を示すコマンドプロンプトのスクリーンショット

③ 完成したレポートをCoursePower上で提出する



提出〆切: 18:30