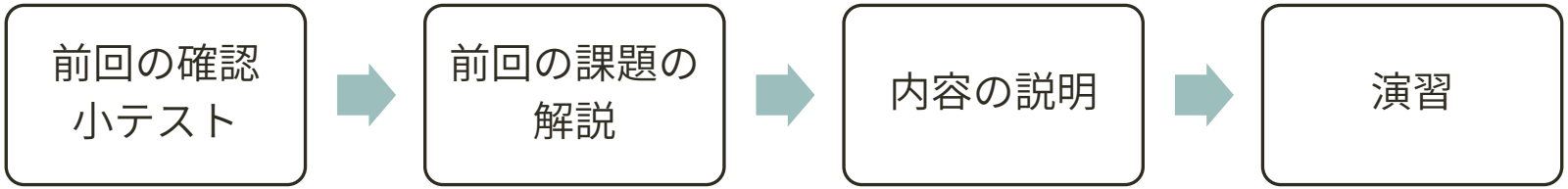
情報処理実習
第13回：ファイル入出力

2017年12月18日(月)
担当：佐藤

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(確認)



基本事項の確認小テスト

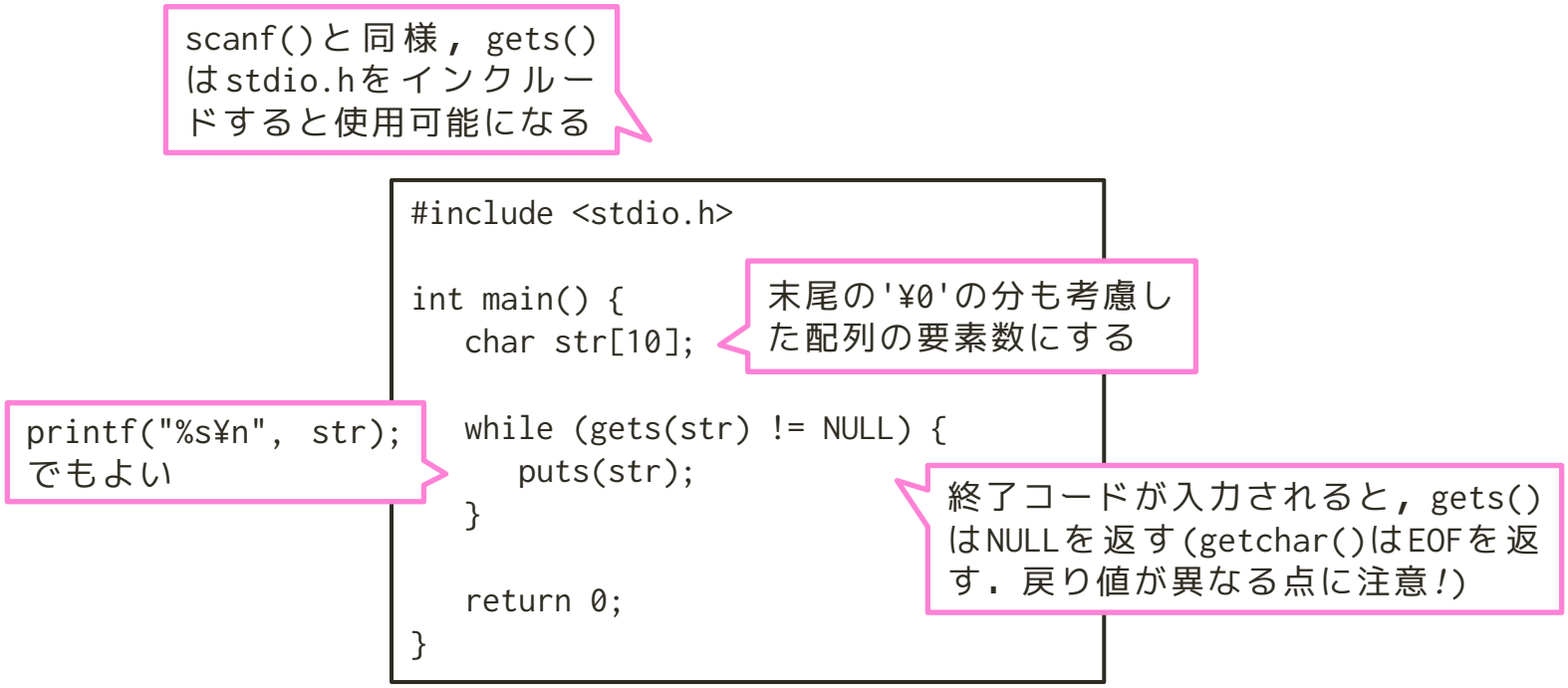
① CoursePowerにログインして，すぐに解答が始められるよう準備する
② 開始の合図があるまで，解答を始めずに待機する

注意

「閉じる」ボタンは決して押さないこと！

前回課題の解説

課題1



- scanf()の%s変換指定子で文字列を読み込む場合，空白文字を読み込むことができない．そのため，この問題では，文字列の読み込みにgets()を使う必要がある

課題2

strcat()を使うために
string.hをインクルード

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

int main() {
    char cStr[100] = "";
    char iStr[20];

    while (gets(iStr) != NULL) {
        strcat(cStr, iStr);
        puts(cStr);
    }

    return 0;
}
```

strcat()は連結先の文字配列の'¥0'の位置に連結元の文字列の先頭の文字を連結する．したがって，連結先の文字配列には'¥0'がなければならない．この条件を満たすため，通常，連結先の文字配列はあらかじめ空文字列にしておく．この処理は初期化で簡単にできる

実は，strcat()の戻り値は連結後の文字列．そのため，
puts(strcat(cStr, iStr));
のようにまとめて1行で書くこともできる

iStrに入力された文字列をcStrに連結し，連結後のcStrを表示することを繰り返す

課題3

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

int main() {
   char lst[10][20] = { "Hawaii", "Egypt", "France", "Russia", "Australia",
                        "Austria", "Germany", "Denmark", "Italy", "Dubai" };
   char str[10];
   int i;

   printf("行きたい旅行先は?¥n");
   scanf("%s", str);
   for (i = 0; i < 10; i++) {
      if (strcmp(lst[i], str) == 0) {
         printf("%sはリストの%d番目です¥n", str, i + 1);
         break;
      }
   }
   if (i == 10) {
      printf("%sはリストにありません¥n", str);
   }

   return 0;
}
```

strcmp()を使うために必要

gets(str);でもよい

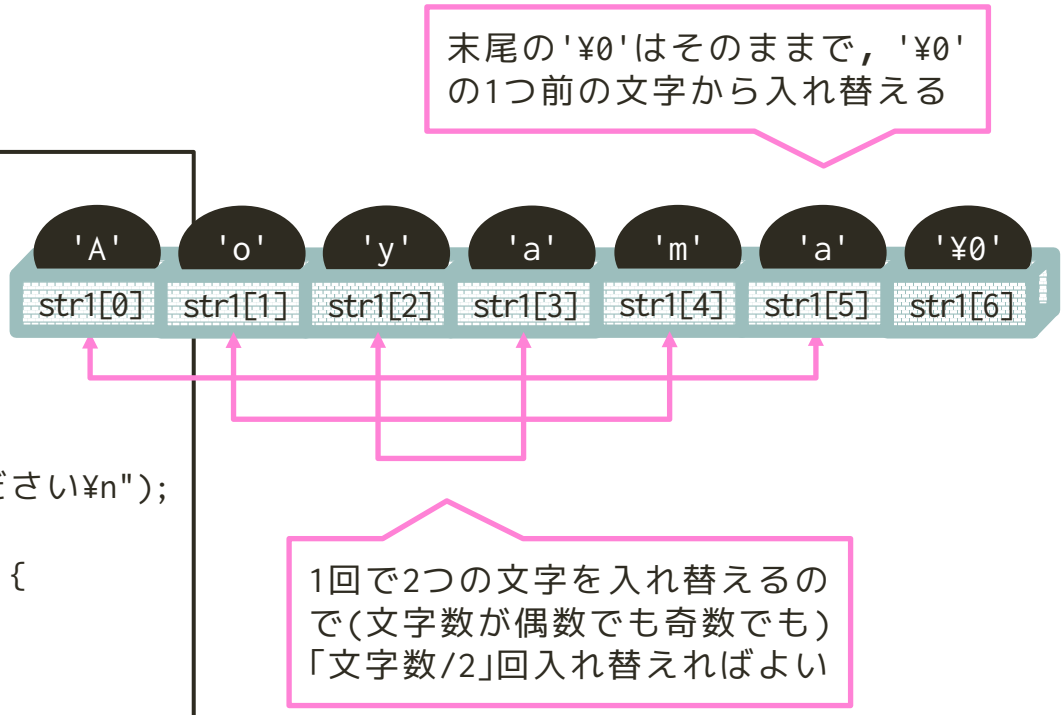
無駄にループしないよう，
表示後すぐループを抜ける

番号調整

無駄にループしないようにするだけでなく，
i==10を「見つからなかった場合」の条件として使えるようになるという効果もある

課題4

```
#include <stdio.h>

int main() {
   char str[7];
   char tmp;
   int i;

   printf("文字列を入力してください¥n");
   scanf("%s", str);
   for (i = 0; i < 7 / 2; i++) {
      tmp = str[i];
      str[i] = str[5 - i];
      str[5 - i] = tmp;
   }
   printf("%s¥n", str);

   return 0;
}
```



ファイルの扱い方

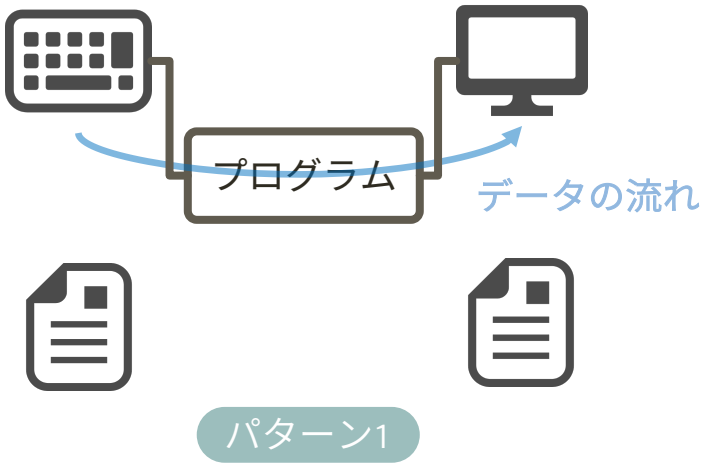
データ処理のパターン

- 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

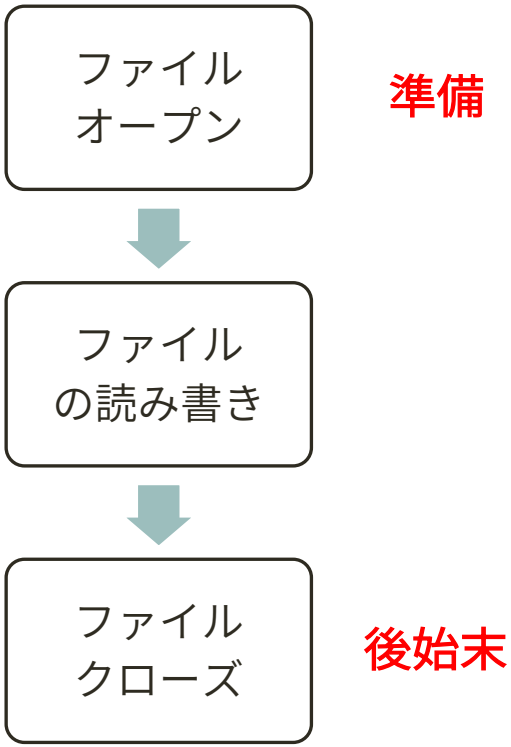
これまでの実習では1のみ行ってきた．今回は2，3，4について学ぶ

これまでのパターン



ファイル入出力の手順

- C言語でファイル入出力処理をする場合，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

ファイルポインタ

- C言語では，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
  - 変数名の前に***(アスタリスク)**を付けて宣言する

文法

```
FILE *変数名;
```

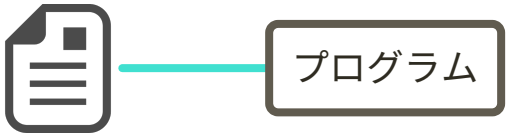
ファイルオープン

- ファイル入出力の準備として，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関数を使う
  - fopen関数の第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，ダブルクォーテーション)で囲む
  - fopen関数を使う前にファイルポインタを宣言しておき，fopen関数の戻り値をファイルポインタに代入する

拡張子(例:「.txt」,「.c」)まで含めて書く必要有り

文法

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



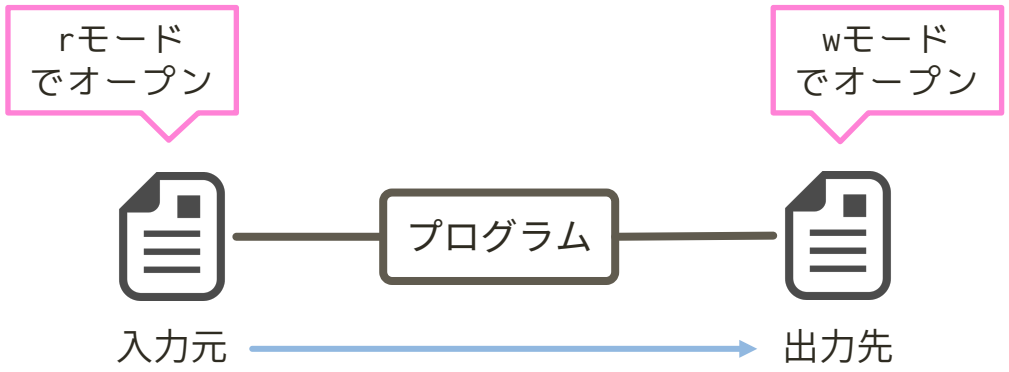
ファイルオープンにより，プログラムとファイルが接続し，データが流れる道(ストリーム)ができる

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関数の第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関数は，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;
}
```

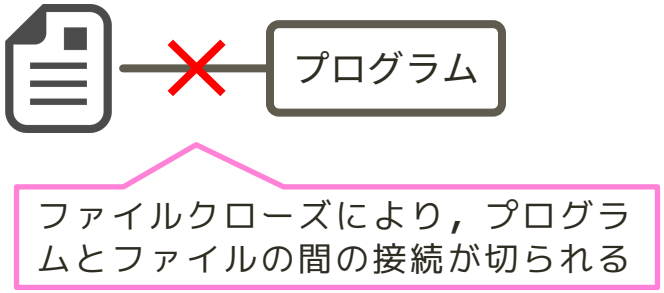
main関数のreturn文の戻り値は，0が正常終了，0以外が異常終了に用いられる

ファイルクローズ

- ファイル入出力の後始末として，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関数を使う

文法

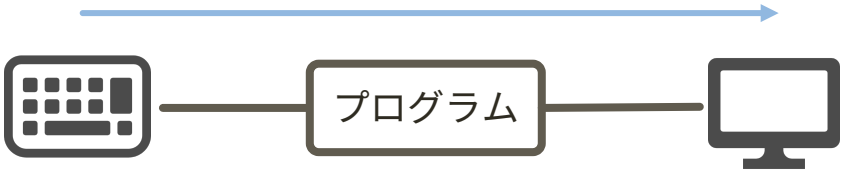
```
fclose(ファイルポインタ);
```



標準入力，標準出力

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



プログラム実行開始時の状況

stdinのオープンによりプログラムと接続するファイルの実体

stdoutのオープンによりプログラムと接続するファイルの実体

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

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関数は，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関数を使う
  - fscanf関数はファイルの中身を**1行だけ**読み込む．続けてもう一度fscanf関数を実行すると，その次の行を読み込む．ファイル中のすべての行を読み込みたい場合，fscanf関数をファイルの行数だけ繰り返し実行する必要がある
  - 最初に実行されたfscanf関数は，ファイルの**先頭行**を読み込む

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)

fprintf()
ファイルへデータを書き込む

- これまで使ってきたprintf関数は，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，**f**printf関数を使う

文法

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

printf関数の第1引数にファイルポインタが追加されただけ！

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()の使用例(1)
成功

```
FILE *fp;
char str1[4], str2[3];
...
fgets(str1, 4, fp);
fgets(str2, 3, fp);
```

'¥n'と'¥0'の2文字分の要素数を含めて宣言

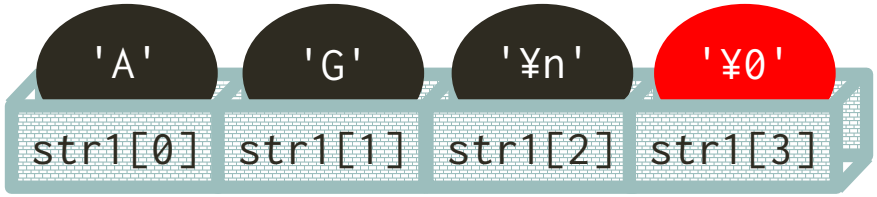
あらかじめファイルオープンとエラー処理を行う

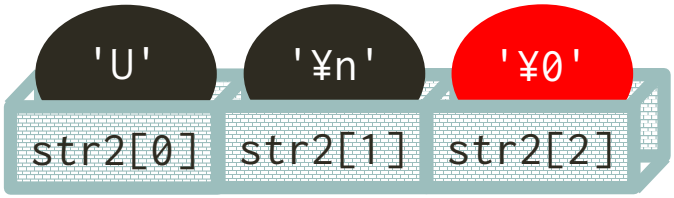
ファイル中の読み込む行の「空白文字と目に見える文字の数+2」以上の整数

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！





ファイル中の読み込む行の空白文字と目に見える文字が漏れなく読み込まれ，かつ，末尾が「'¥n''¥0」ならば成功

fgets()の使用例(2)
失敗

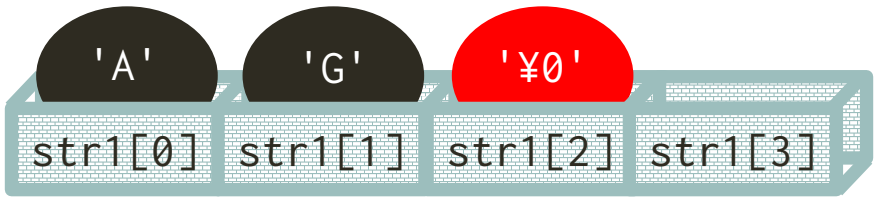
```
FILE *fp;
char str1[4], str2[3];

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

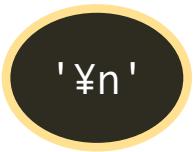
間違い！

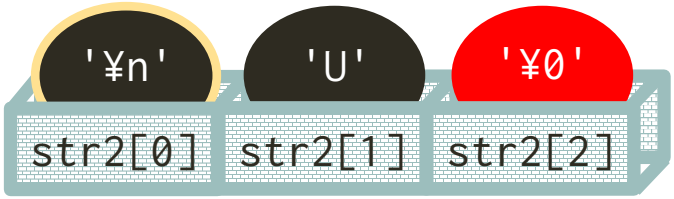
読み込むファイル

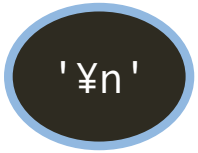
AG¥n
U¥n

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！

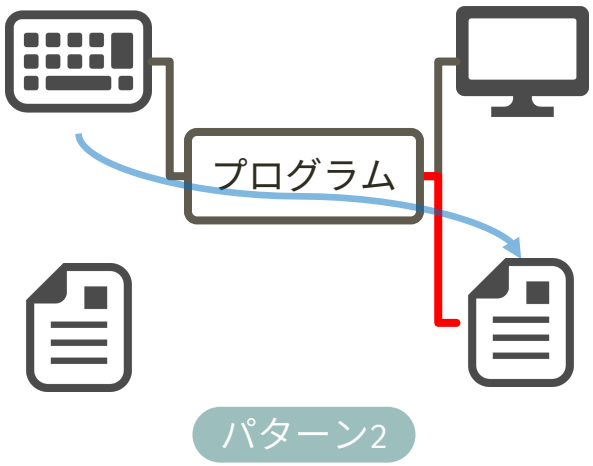
サンプルプログラム

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

```c
#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;
   }

   /* 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;
}
```




```c
fscanf(stdin, "%d %d %d", &v[0], &v[1], &v[2]);
```

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

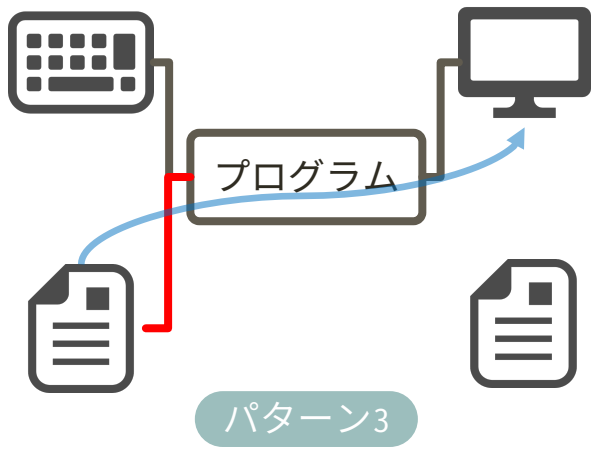
```
10 20 30⏎
続行するには何かキーを押してください. . .
```

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

```c
#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;
}
```



パターン3

data.txt

```
10 20 30
```

```c
fprintf(stdout, "%d %d %d", &v[0], &v[1], &v[2]);
```

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あらかじめ**「data.txt」を入れておく**こと！

実行結果

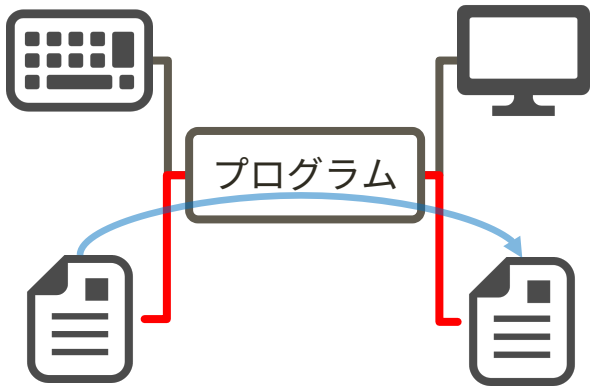
```
10 20 30
続行するには何かキーを押してください. . .
```

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

```
#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;
   }
```

(右へ続く)

```
   /* 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;
}
```



パターン4

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あらかじめ**「data.txt」を入れておく**こと！

実行結果

```
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

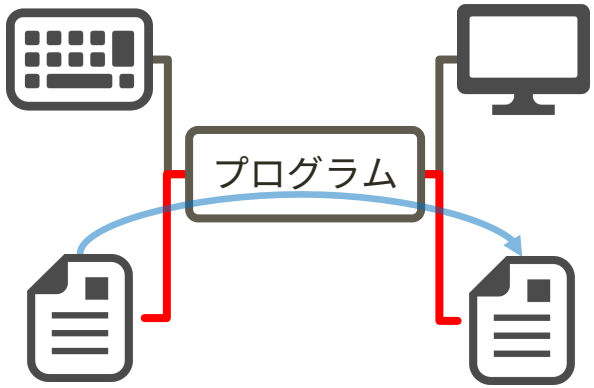
```c
#include <stdio.h>

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;
   }
```

読み込むファイル中の1行あたりの「空白文字と目に見える文字の最大数+2」以上の要素数が必要

(右へ続く)

```c
   /* 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



パターン4

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あらかじめ**「data.txt」を入れておく**こと！

実行結果

```
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(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，サンプルプログラム(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ことに注意すること！

課題

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13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