



情報処理演習II

No. 12

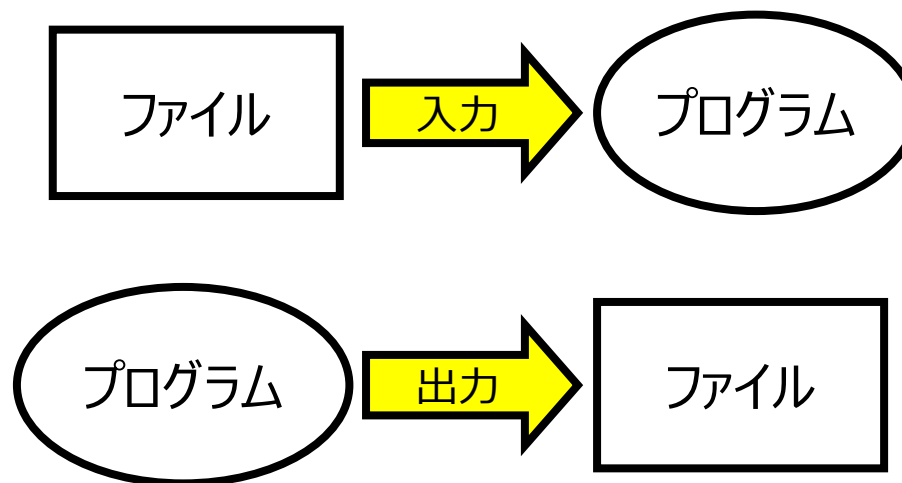
2023. 10. 31

芝浦工業大学 システム理工学部 機械制御システム学科

担当：桑原

ファイル操作

- 複雑な用途のために作成したプログラムでは、ファイル操作が必要になる
(プログラム本体だけでは、様々な用途に対応できない)
 - ユーザ設定を行うプログラム (ブラウザ, メーラ, ゲーム等)
 - 大量のデータを扱うプログラム (数値シミュレーション等)
- ファイルの読み込み
 - データの読み込み
 - 設定の読み込み
- ファイルに書き込み
 - データの保存



ファイル操作の手順

1. **ファイルポインタ**を準備する

例： `FILE *fp;`

→ ファイルポインタ `fp` を宣言。先頭の「`*`」に注意

2. ファイルを**オープン**する

例： `fp = fopen("data.txt", "w");`

→ ファイル `data.txt` を書き込みモード（`"w"`）でオープンする。

以後、このファイルはファイルポインタ `fp` で指定される

3. ファイルのデータを読み込む/ファイルにデータを書込む

例： `fprintf`関数, `fscanf`関数などを利用

4. ファイルを**クローズ**する

例： `fclose(fp);`

→ ファイルポインタ `fp` を閉じて、ファイルへのアクセスを終了

サンプルプログラム

サンプルプログラム 12.1

```
#include <stdio.h>

int main(void)
{
    FILE *fp;                /* ファイルポインタ fp の宣言 */

    fp = fopen("test.txt", "r"); /* ファイル test.txt を読み込みモードでオープン */
    if (fp == NULL) {          /* 「何らかの理由」でファイルオープンに失敗した場合 */
        printf("指定されたファイルが開けませんので、プログラムを終了します。 ¥n");
        return(-1);           /* 異常をOSに伝えて実行終了 */
    } else {
        printf("指定されたファイルをオープンしました。 ¥n"); /* ファイルオープンに成功 */
    }

    /* ここにファイルへの記載内容を書く */

    fclose(fp);               /* ファイルをクローズ */
    printf("ファイルをクローズしました。 ¥n");
    return 0;
}
```

ファイルポインタの宣言

```
FILE *ファイルポインタ名;
```

- ファイルポインタ = FILE型へのポインタ
 - FILE型はデータ型のようなもの
 - ファイルの読書き・クローズを行う際には、ファイル名ではなく**ファイルポインタを指定**する
- 例： FILE *fp; (意味：ファイルポインタ fp を宣言)
- 同時に複数のファイルを扱う場合には、必要な分だけ宣言する
例： FILE *fp1, *fp2; (2つのファイルポインタを宣言)

ファイルのオープン

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

- ファイルのオープン： OSに「これから（ファイル名）を（モード）目的で使しますよ」と宣言
- ファイル名は文字列
 - 文字列は「"」で囲む 例： "test.txt", "data.txt", "sample.dat" など
- モード
 - "r" : 読込, "w" : 書込, "a" : 追加書込, など
- 例： `fp = fopen("sample.dat", "r");`
(意味：ファイル sample.dat を読込モードでオープンし、以降ではこのファイルをファイルポインタ fp で表すことにする)
- 何らかの理由でファイルをオープンできなかった場合には、
NULL ポインタ（空のポインタ）がファイルポインタに設定される。
→ エラー処理に利用可能

テキストファイル（参考）

- テキストファイル = 人間が読める文字の集まり、つまり文字列 集合。
ファイルの最後には「**EOF**（end of file）」と呼ばれる特殊な文字が置かれている。
また、**ファイルポインタがこれを読むと、その値は-1となる。**

a) 人間から見たテキストファイルのイメージ：

```
This is a pen.  
World.  
Bye!
```

b) 計算機からみたコンパイル時のイメージ：

'T'	'h'	'i'	's'	' '	'i'	's'	' '	'a'	' '	'p'	'e'	'n'	'.'	'\n'	'W'	'o'
'r'	'l'	'd'	'.'	'\n'	'B'	'y'	'e'	'!'	EOF							

ファイルのクローズ

```
fclose(ファイルポインタ名);
```

- ファイルのクローズ
= OSに「（ファイルポインタ名）が指定するファイルの操作を終了しますよ」と宣言
- ファイルをオープンしたら必ずクローズする。
これを 怠るとOSが異常動作する可能性がある
- 例： `fclose(fp);`


```
fprintf(ファイルポインタ名, "書式", 変数名など);
```

- (ファイルポインタ名) が指定するファイルに、書式の内容を書込む
 - printf の「画面に出力」が「ファイルに書込み」に変わっただけ！
 - ファイルは書込み可能でオープンしておくこと！

• 例 : fprintf(fp, "data = %f¥r¥n", x);

x は float 型変数で、値は 10 とする。

「¥r¥n」でprintfの
「¥n」と同じ効果

→ fp の指定するファイルに「data =10 (改行)」を書込む
(printf関数の時の文字列の出力先が、スクリーンからファイルに変わっただけ)

ファイルへの文字列書込み

[手順1] 文字列変数の準備： 文字型の配列変数を宣言

例) `char name[20];`

→ 半角20文字分を記憶できる文字列変数nameを宣言

- (注意) 1. 文字変数は、半角文字1文字しか記憶できないが、これを配列にすることで文字列を取り扱える。
2. 配列要素数は、事前に十分な文字数を確保しておく。

[手順2] 文字列データの書込み

例) `fprintf(fp, "%s", name);`

→ ファイルポインタfpで指定したファイルに、変数nameに入っている文字列を
文字列 (%s) として書込む

- (注意) 文字配列変数は、scanfの場合と同様にprintfの場合でもポインタを指定する
(= 配列変数名そのもの) 。

```
fscanf(ファイルポインタ名, "書式", &変数名);
```

- ファイルポインタ名が指定するファイルの内容を読み込み、それを変換書式に従って変数に代入する。
 - scanf関数 の「キーボード入力」が「ファイルからの読み込み」になっただけ。
 - ファイルは読み込み可能でオープンしておくこと。
 - 読取りに成功すると、読込んだデータの個数が戻り値となる。
(EOF (ファイルの終端部) なら, 戻り値として -1 を返す)
- 例 : fscanf(fp, "%lf %lf %lf", &x, &y, &z);
→ fp の指定するファイルを読み込み、3つの浮動小数点数データをdouble型変数 x, y, z にそれぞれ代入。読取り行は実行の度に自動的に次の行に移る。

ファイルからの文字列読み込み

[手順1] 文字列変数の準備： 文字型の配列変数を宣言

例) `char name[20];`

→ 半角20文字分を記憶できる文字列変数nameを宣言

- (注意) 1. 文字変数は、半角文字1文字しか記憶できないが、これを配列にすることで文字列を取り扱える。
2. 配列要素数は、事前に十分な文字数を確保しておく。

[手順2] 文字列データの読み込み

例) `fscanf(fp, "%s", name);`

→ ファイルポインタfpで指定したファイルから、変数nameに文字列（%s）としてデータを読み込む。（配列変数名はポインタでもあったことに注意）

サンプルプログラム 各自で書いて実行してみてください。

サンプルプログラム 12.2

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    double x, y, z;

    fp = fopen("data.dat", "r");
    if (fp == NULL) { /* 何らかの理由でファイルオープンに失敗した場合の処理 */
        printf("指定されたファイルが開けませんので、プログラムを終了します。 ¥n");
        return(-1); /* 異常をOSに伝えて実行終了 */
    } else {
        printf("指定されたファイルをオープンしました。 ¥n");
    }

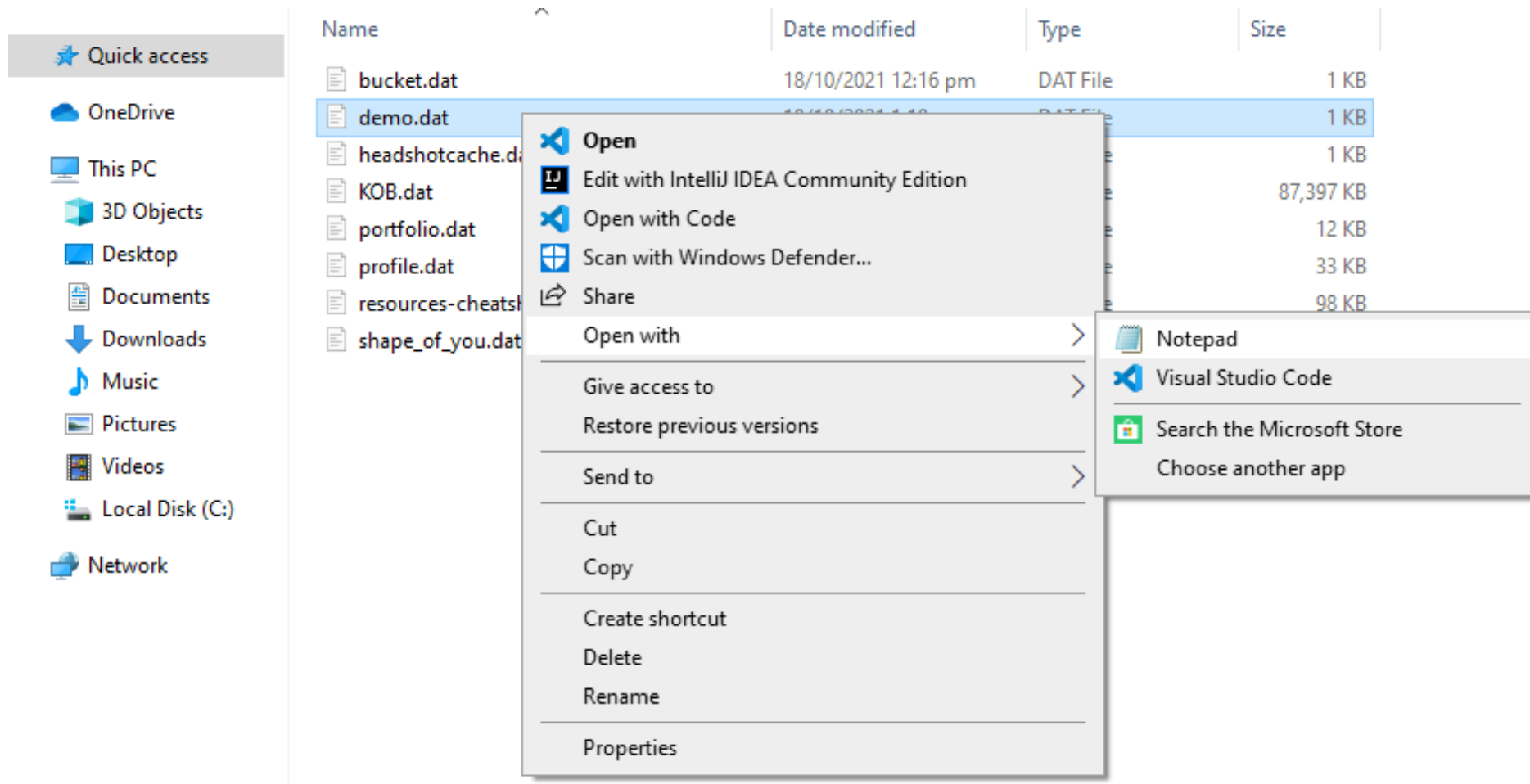
    fscanf(fp, "%lf, %lf, %lf", &x, &y, &z); /* 読み込みデータ数のチェック(半角カンマ区切り) */

    fclose(fp);
    printf("ファイルをクローズしました。 ¥n");

    printf("読み込んだデータの総和は %f です。 ¥n", x + y + z);
    return 0;
}
```

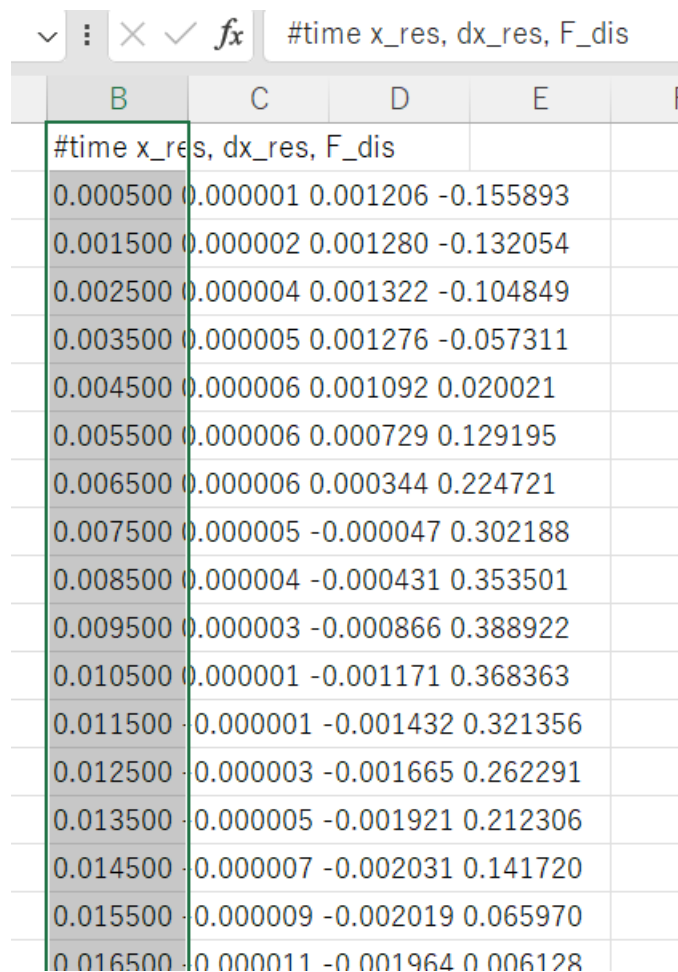
datファイルのデータを表示する方法（１）

1. メモ帳などで開く



datファイルのデータを表示する方法（2）

- EXCELにコピー



	B	C	D	E
	#time x_res, dx_res, F_dis			
	0.000500	0.000001	0.001206	-0.155893
	0.001500	0.000002	0.001280	-0.132054
	0.002500	0.000004	0.001322	-0.104849
	0.003500	0.000005	0.001276	-0.057311
	0.004500	0.000006	0.001092	0.020021
	0.005500	0.000006	0.000729	0.129195
	0.006500	0.000006	0.000344	0.224721
	0.007500	0.000005	-0.000047	0.302188
	0.008500	0.000004	-0.000431	0.353501
	0.009500	0.000003	-0.000866	0.388922
	0.010500	0.000001	-0.001171	0.368363
	0.011500	0.000001	-0.001432	0.321356
	0.012500	0.000003	-0.001665	0.262291
	0.013500	0.000005	-0.001921	0.212306
	0.014500	0.000007	-0.002031	0.141720
	0.015500	0.000009	-0.002019	0.065970
	0.016500	0.000011	-0.001964	0.006128

- EXCELのメニューバー
「データ」→「区切り位置」



区切り位置指定ウィザード - 1 / 3

選択したデータは区切り文字で区切られています。

[次へ] をクリックするか、区切るデータの形式を指定してください。

元のデータの形式

データのファイル形式を選択してください：

☒ コンマやタブなどの区切り文字によってフィールドごとに区切られたデータ(D)

☐ スペースによって右または左に揃えられた固定長フィールドのデータ(W)

選択したデータのプレビュー：

1	#time x_res, dx_res, F_dis
2	0.000500 0.000001 0.001206 -0.155893
3	0.001500 0.000002 0.001280 -0.132054
4	0.002500 0.000004 0.001322 -0.104849
5	0.003500 0.000005 0.001276 -0.057311
6	0.004500 0.000006 0.001092 0.020021

キャンセル < 戻る(B) 次へ(N) > 完了(E)

datファイルのデータを表示する方法（3）

区切り位置指定ウィザード - 2 / 3

フィールドの区切り文字を指定してください。[データのプレビュー] ボックスには区切り位置が表示されます。

区切り文字

☐ タブ(I)

☐ セミicolon(M)

☒ コンマ(C)

☒ スペース(S)

☐ その他(Q):

☒ 連続した区切り文字は 1 文字として扱う(R)

文字列の引用符(Q): "

データのプレビュー(P)

#time	x_res	dx_res	F_dis
0.000500	0.000001	0.001206	-0.155893
0.001500	0.000002	0.001280	-0.132054
0.002500	0.000004	0.001322	-0.104849
0.003500	0.000005	0.001276	-0.057311
0.004500	0.000006	0.001092	0.020021

キャンセル < 戻る(B) 次へ(N) > 完了(E)

	B	C	D	E
#time	x_res	dx_res	F_dis	
0.0005	0.000001	0.001206	-0.15589	
0.0015	0.000002	0.00128	-0.13205	
0.0025	0.000004	0.001322	-0.10485	
0.0035	0.000005	0.001276	-0.05731	
0.0045	0.000006	0.001092	0.020021	
0.0055	0.000006	0.000729	0.129195	
0.0065	0.000006	0.000344	0.224721	
0.0075	0.000005	-4.7E-05	0.302188	
0.0085	0.000004	-0.00043	0.353501	
0.0095	0.000003	-0.00087	0.388922	
0.0105	0.000001	-0.00117	0.368363	
0.0115	-1E-06	-0.00143	0.321356	
0.0125	-3E-06	-0.00167	0.262291	
0.0135	-5E-06	-0.00192	0.212306	
0.0145	-7E-06	-0.00203	0.14172	
0.0155	-9E-06	-0.00202	0.06597	
0.0165	-1.1E-05	-0.00196	0.006128	

課題 1 2

- 今回作成したソースプログラムをGoogleDriveの「課題提出¥課題12」に提出してください。
本日、2限終了時刻を〆切とします。
 - 課題12-1：以下の機能を満たすプログラムを作成せよ。
 - データの出力ファイル名を“data.dat”とする。
 - $t=0$ 秒から0.01秒ごとインクリメントさせ $t=2$ 秒まで繰り返し処理を行う。
 - main関数でfor文を使い、以下の処理を実行する。
 - t を入力する。
 - 小数型変数out1に振幅1、周期1秒のsin波の値を代入する。
 - 小数型変数out2に振幅2、周期1秒のcos波の値を代入する。
 - 出力ファイルに時刻 t , out1, out2を出力する。
 - 出力ファイルを閉じる。
- * #include <math.h>を使って $\sin(\theta)$ を計算する。ただし、 θ はラジアン単位。

課題 1 2

- 今回作成したソースプログラムをGoogleDriveの「課題提出¥課題12」に提出してください。
本日、2限終了時刻を〆切とします。
 - 課題12-2：以下の機能を満たすプログラムを作成せよ。
 - 新しく出力用ファイルを生成する。“data2.dat”
 - 課題12-1で生成したdata.datから時刻t, 出力out1, out2を読み込む。
 - * EOFまで読み込むとデータ数が把握できる。
 - for文で時刻tのデータ数分だけ以下を繰り返す
 - 読み込んだ時刻tを2倍し、時刻t2に代入する。
 - 時刻t2を入力とし、小数型変数out3に振幅1、周期1秒のsin波の値を計算し代入する。
 - 出力ファイルに時刻t2, out3を出力する。
 - 出力ファイル(“data2.dat”)を閉じる。
- * #include <math.h>を使ってsin(theta)を計算する。ただし、thetaはラジアン単位。

情報処理II・情報処理演習II 中間試験要領

情報処理II

- 11/14(火) 1限 プログラミング教室
- 持ち込み不可。
- 範囲：初回講義で説明。ScombZの情報処理IIのLMSから確認可能。

情報処理演習II

- 11/14(火) 2限 プログラミング教室
- 講義資料の印刷物、PCの持ち込み可。
- C言語の開発環境、EXCEL・WORD（またはそれに準じるソフト）がインストールされているPCを使用すること。プログラミング教室のPCを使う場合は、環境を事前に確認しておくこと。
- 会話・携帯・インターネットでの調べものは禁止（発見次第、中断）
- 質問は教員・TAにのみ可能
- 範囲：第1～12回の内容
- 2限終了前に提出完了した人は速やかに退出すること。
- **提出物：ソースコードおよび指定WORDファイル**
- **提出先：ScombZ**