C プログラミング 入門

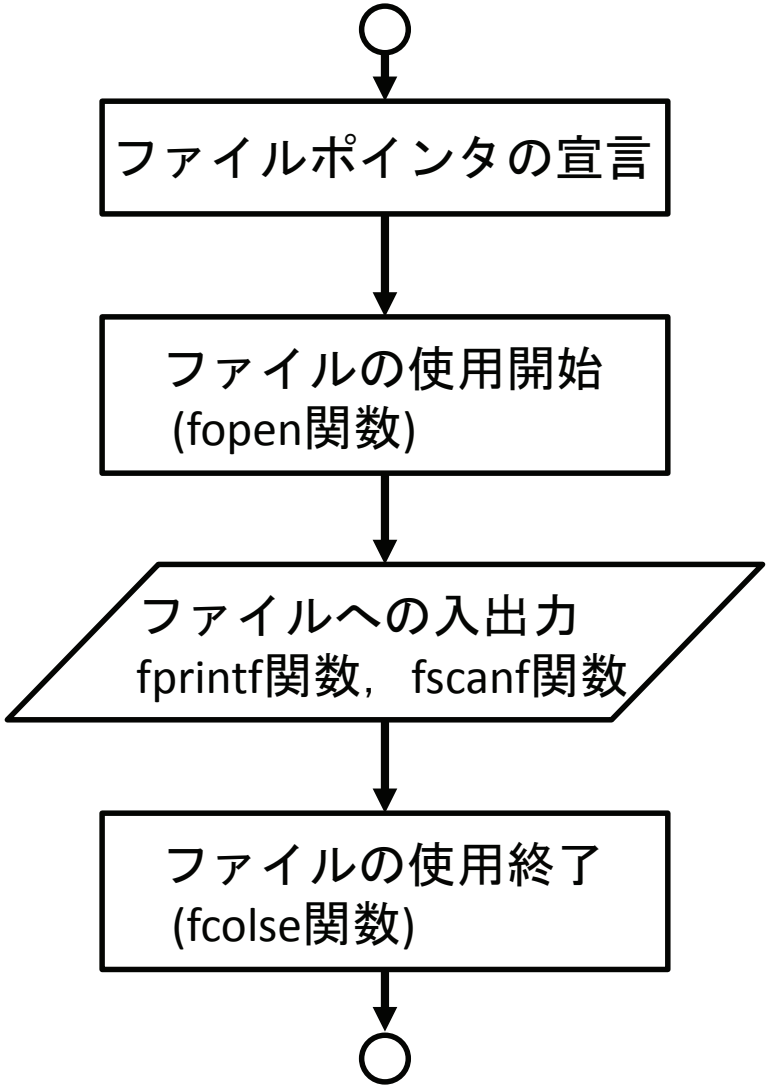
— ファイル入出力 —

早稲田大学

本日の目標

- ファイルの読み書きができる.
  - ファイルを開く（fopen 関数）
  - ファイルを閉じる（fclose 関数）
  - ファイルに書き込む（fprintf 関数）
  - 制御文字・printf 書式まとめ
  - ファイルから読み込む（fscanf 関数）
- エラー処理
  - exit 関数

ファイルの取り扱い方法



FILE *ファイルポインタ名

fopen(“ファイル名”,”モード”)

fprintf(ファイルポインタ名,書式,値リスト)

fscanf(ファイルポインタ名,書式,値リスト)

fclose(ファイルポインタ名)

ファイルポインタ

ファイルポインタ

```
FILE *変数名;
```

- FILE *で宣言された変数を' ファイルポインタ' と呼ぶ
- FILE は stdio.h に定義されているが，詳しい内容は特に知る必要ない

ファイルを開く・閉じる

fopen 関数　fopen("ファイル名","モード")

```
FILE *file;     /* ファイルポインタ file の宣言 */
file = fopen("ファイル名","モード");
```

- ファイルをファイル名を指定して開く場合に使用
- モードは例えば以下を指定する

モード	意味
w	ファイルを新しく作成し，書き込めるようにする（既にそのファイルが存在すれば，上書きされる）
r	既存ファイルを読み込み専用に開く

- モードが r では開けなければ NULL を返す（→エラー検出可能）

fclose 関数

- ファイルを使用し終えたら，ファイルを閉じる

```
fclose（ファイルポインタ）;
```

ファイルに書き込む

fprintf 関数

fprintf(ファイルポインタ，書式，出力したい変数);

- ファイルを"w"モードで作成し，書き込む（ファイルが既にある場合は，上書きされるので注意）
- ファイル作成例

```
int a=10;
FILE *file=fopen("test","w");          /*test という名前のファイルを作成*/
fprintf(file, "%d¥t",a);              /*ファイルに「10 タブ」を書き込む*/
fprintf(file, "%d",a+1);               /*ファイルに「11」を書き込む*/
fprintf(file, "%d¥n",a*2);            /*ファイルに「20 改行」を書き込む*/
fclose(file);
```

制御文字・printf 書式まとめ

- C 言語の文字列では，¥で始まる文字は特別な意味を持つ
  （これらはコンパイルされると一文字に置き換わる【参考】）

¥n	¥t	¥"	¥¥
改行	タブ	「"」	「¥」

- printf は第一引数で指定した文字列を画面に表示する
- printf 書式で%で始まる文字は，指定する値に置き換わる

書式例	動作	書式例	動作
%d	整数の 10 進数表示	%x	整数の 16 進数表示
%5d	5 桁未満ならスペースを挿入し 5 桁にする	%05d	5 桁未満なら「0」を挿入し 5 桁にする
%f	実数の 10 進数表示	%.2f	小数点以下を 2 桁まで表示
%5.2f	5 文字未満ならスペースを挿入し 5 文字にする（小数点含む）	%+5.2f	正の値の場合は,「+」をつける
%p	ポインタを表示	%e	実数を「1.23e-4」の形式で表示
%c	1 バイト文字を表示	%s	文字列を表示
%%	「%」を表示.		

例題 1

例題 1：以下のプログラムを作成せよ

- ファイル名は「kukuout.c」とする
- テキストファイル「kuku」を作成して，タブ区切りで九九の表を書き込め
- fprintf で"%2d¥ t" と"¥ n" を使うこと
- いきなりファイルに書き込むと正しく書き込めているかどうかのチェックが面倒なので，最初は画面に表示させてチェックする

```
 1   2   3   4   5   6   7   8   9
 2   4   6   8  10  12  14  16  18
 3   6   9  12  15  18  21  24  27
 4   8  12  16  20  24  28  32  36
 5  10  15  20  25  30  35  40  45
 6  12  18  24  30  36  42  48  54
 7  14  21  28  35  42  49  56  63
 8  16  24  32  40  48  56  64  72
 9  18  27  36  45  54  63  72  81
```

ファイルから読み込む

fscanf 関数

- scanf でキーボードから読み込むのと同じようにファイルから読み込み可能
- ファイルは fopen が返したファイルポインタで指定できる
- ホワイトスペース（半角空白，タブ，改行）は自動的にスキップされる
- 戻り値で，読み込んだ個数を返す（エラー検出可能）

  fscanf(ファイルポインタ，書式，ポインタリスト);

- ファイル名を間違って変更してしまわないよう"r"モードで開いて読み込む

```
int a; double b;
FILE *file=fopen("test","r");    /*test ファイルを読み込むモードで開く*/
fscanf(file, "%d",&a);           /*書かれている整数を読みとり，a に代入*/
fscanf(file, "%lf",&b);          /*書かれている実数を読みとり，b に代入*/
fclose(file);
```

補足

- C 言語は 3 つのファイルポインタが宣言されている
- 標準入力 stdin $\rightarrow$ キーボード
  fprintf(stdin,"$\cdots$",$\cdots$) $\Leftrightarrow$ scanf("$\cdots$",$\cdots$)
- 標準出力 stdout $\rightarrow$ 画面 (./a.out を実行したあとの行)
  fprintf(stdout,"$\cdots$",$\cdots$) $\Leftrightarrow$ printf("$\cdots$",$\cdots$)
- 標準エラー出力 stderr $\rightarrow$ 画面 (./a.out を実行したあとの行)
- stdout と stderr の違いは UNIX 系 OS のリダイレクトに関係

エラー処理

exit 関数

- プログラムは，予定外のことが起こった場合は終了させるのが安全（終了せずにうまく実行させることは一般に難しい）
- main 関数を最後まで実行することなく，プログラムを強制終了させることができる
- #include<stdlib.h>により使えるようになる
- 通常，引数に 0 以外の値（異常終了を示す）を指定する（正常終了の場合は main 関数の最後で，return 0; としたように 0 を返す）

```
exit(1);
```

- エラーを特定しやすいよう，exit を実行する前に終了理由を画面に出す

```
fprintf(stderr,"エラーメッセージ");
```

エラー処理

ファイルが開けなかった場合の処理例

```
FILE *file = fopen("test","r");
if(file==NULL){                                  /*ファイルが開けなかった*/
    fprintf(stderr, "cannot open file 'test'");          /*エラー*/
    exit(1);                                      /*プログラムの強制終了*/
}
```

ファイルから期待通りに読み込めなかった場合の処理例

```
double a; int scannum;
FILE *file ;
...
scannum = fscanf(file,"%lf",&a);                  /*実数を一つ読み込む*/
if(scannum!=1){                                  /*データが読み込めなかった*/
    fprintf(stderr, "cannot read file");                 /*エラー*/
    exit(1);                                      /*プログラム強制終了*/
}
```

- ファイルを閉じる（fclose）等は exit がやってくれるため，不要

例題 2

例題 2：以下のプログラムを作成せよ

- ファイル名は「kukuin.c」とする
- 「kuku」を読み込んで，正誤チェックして画面に表示せよ
- 値が正解の場合は,「-」を，間違っている場合は「X」を数字の後に表示せよ

```
1-   2-   3-   4-   5-   6-   7-   8-   9-
2-   4-   6-   8-  10-  12-  14-  16-  18-
3-   6-   9-  12-  15-  18-  21-  24-  27-
4-   8-  12-  16-  20-  24-  28-  32-  36-
5-  10-  15-  20-  25-  30-  36X  40-  45-
6-  12-  18-  24-  30-  36-  42-  48-  54-
7-  14-  11X  28-  35-  42-  49-  56-  63-
8-  16-  24-  32-  40-  48-  56-  64-  72-
9-  18-  27-  36-  45-  54-  63-  72-  81-
```

- 「kuku」をわざと間違った値にして動作チェックせよ
- ファイルが開けなかった場合，値が読み込めなかった場合はエラーメッセージを出して exit する

本日のまとめ

- ファイルの読み書きができる.
  - ファイルを開く（fopen 関数）
  - ファイルを閉じる（fclose 関数）
  - ファイルに書き込む（fprintf 関数）
  - 制御文字・printf 書式まとめ
  - ファイルから読み込む（fscanf 関数）
- エラー処理
  - exit 関数