12 ファイル入出力

12–1 ファイル入出力 — fopen() と fclose() —

Cプログラムで操作するファイルは通常はテキストファイルで、
前から順に処理される。

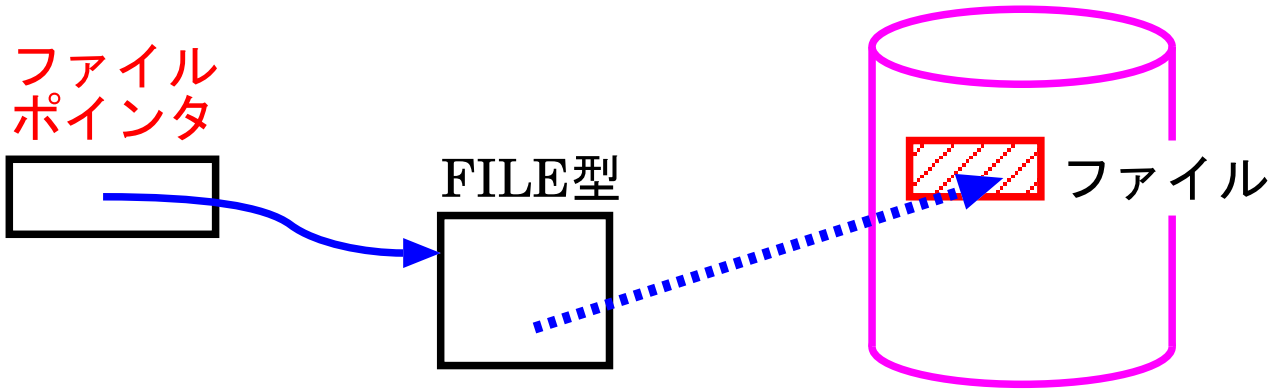
例えば、
ファイルからデータを入力する場合は、...
また、ファイルへデータ出力する場合は、...

⇨ ファイル操作の間ファイルのどの場所を現在処理しているかの情報を常にどこかに保持しておく必要があり、この情報も含めてファイル操作を快調に行うための情報を維持しておく必要がある。

しかし、ディスク上のファイルに直接アクセスするという操作は、システム管理の問題と関わって来るので、一般ユーザには許されていない。

そこで、C言語においては 、

- プログラムからの要求に応じて、
  言語処理系/OSがこれらのファイル操作に必要/有用な情報をFILE
  というデータ型名の付いた構造体の中に詰め込み、
- プログラムの中でこのFILE型構造体へのポインタ (i.e. 所在番地) を
  保持する、

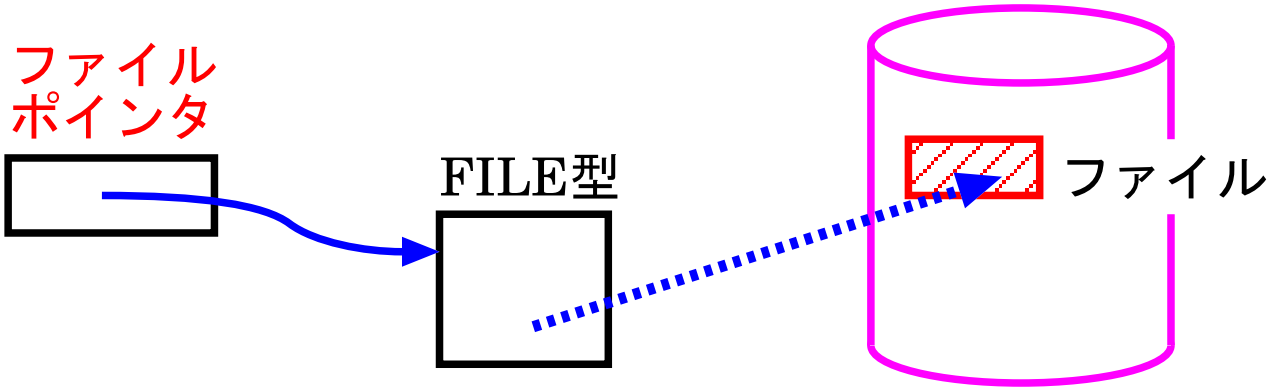


そして

- プログラムの中では、FILE型構造体へのポインタを明示することに
  よって、間接的に操作目的のファイルを指定する、

という仕組みが取られている。

このFILE型構造体へのポインタのことをファイルポインタと言う。



また、
操作を始めたいファイルに関するFILE型構造体を言語処理系/OSに作ってもらってファイル操作の準備を行うことを、ファイルをオープンすると言い、逆に操作の終ったファイルのFILE型構造体の領域を解放してファイル操作の後始末を行うことをファイルをクローズすると言う。

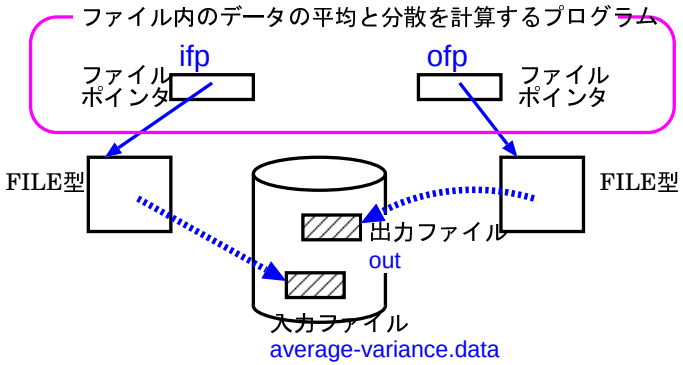
例題12. 1 (ファイル内のデータの平均と分散)　例題9.1と同じ様に、50個の実数データ $x_0, x_1, x_2, \cdots, x_{49}$ を読み込み、それらの平均 μ と分散 V を定義式

$$\mu = (x_0 + x_1 + x_2 + \cdots + x_{49})/50$$

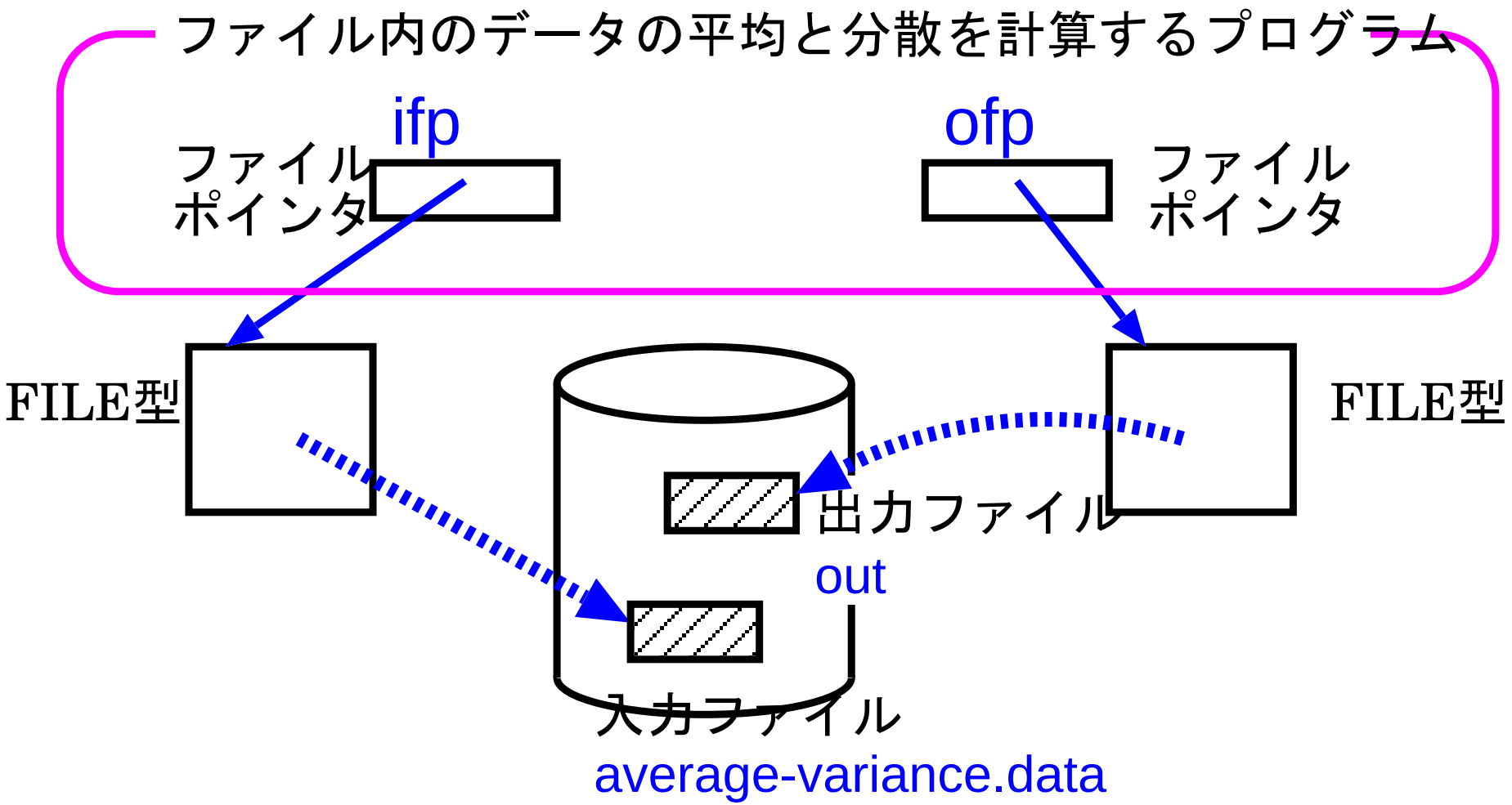
$$V = \sum_{i=0}^{49} (x_i - \mu)^2/50$$

に従って求めて出力するCプログラムを作成せよ。但し、ここでは入力データは average-variance.data という名前のファイルから読み込み、出力データは out というファイルに書き出すことにする。

(考え方)　入力ファイルに繋がるファイルポインタ，出力ファイルに繋がるファイルポインタの領域が必要である。



拡大→次ページ


ファイル内のデータの平均と分散を計算するプログラム
ifp
ファイル
ポインタ
ofp
ファイル
ポインタ
FILE型
FILE型
出力ファイル
out
入力ファイル
average-variance.data

例題12. 1 (ファイル内のデータの平均と分散)　例題9.1と同じ様に、50個の実数データ $x_0, x_1, x_2, \ldots, x_{49}$ を読み込み、それらの平均 μ と分散 V を定義式

$$\mu = (x_0 + x_1 + x_2 + \cdots + x_{49})/50$$

$$V = \sum_{i=0}^{49} (x_i - \mu)^2/50$$

に従って求めて出力するCプログラムを作成せよ。但し、ここでは入力データは average-variance.data という名前のファイルから読み込み、出力データは out というファイルに書き出すことにする。

(考え方)　入力ファイルに繋がるファイルポインタ, 出力ファイルに繋がるファイルポインタの領域が必要である。

これらのものが用意されていれば、基本的なデータ処理の流れは例題9.1と同じで良いので、次の4点に注意して例題9.1のプログラムに少し手を加えるだけである。

(考え方)　入力ファイルに繋がるファイルポインタ, 出力ファイルに繋がるファイルポインタの領域が必要である。

これらのものが用意されていれば、基本的なデータ処理の流れは例題9.1と同じで良いので、次の4点に注意して例題9.1のプログラムに少し手を加えるだけである。($\Longrightarrow$10.8節を参照)

- データ処理の前にライブラリ関数 `fopen()` を用いて2つのファイルをオープンする必要がある。
- データ処理の後にライブラリ関数 `fclose()` を用いて2つのファイルをクローズする必要がある。
- 指定されたファイルからデータを入力するので、`scanf()` じゃなく `fscanf()` を用いる。
- 指定されたファイルへデータを出力するので、`printf()` じゃなく `fprintf()` を用いる。

(プログラミング)

例題9.1で考えた配列や変数の他に、
ファイルポインタを保持のためにifp, ofp という名前の変数を
用意して、プログラムを構成した。

```
[motoki@x205a]$  nl average-variance-io-through-files.c
     1 /* 50個の実数データ x0, x1, x2, ... , x49 の平均 mu と
     2 /* 分散 $V$ を定義式
     3 /*    mu = (x0+x1+x2+ ...  + x49)/50
     4 /*     V = (x0-mu)^2 + (x1-mu)^2 + ... + (x49-mu)^2 / 50
     5 /* に従って求め、それらの値を出力するプログラム
     6 /* 但し、ここでは入力データは average-variance.data とい.
     7 /* 名前のファイルから読み込み、出力データは out というフ..*/
     8 /* イルに書き出すことにする。

     9 #include <stdio.h>
    10 #include <stdlib.h>
```

```
11 int main(void)
12 {
13   int      i;
14   double   x[50], ave, var;
15   FILE     *ifp, *ofp;

16   if ((ifp=fopen("average-variance.data", "r"))
                                          == NULL) {
17     printf("ファイルをオープン出来ません: average-varianc
18     exit(1);
19   }
20   if ((ofp=fopen("out", "w")) == NULL) {
21     printf("ファイルをオープン出来ません: out\n");
22     exit(1);
23   }
24
```

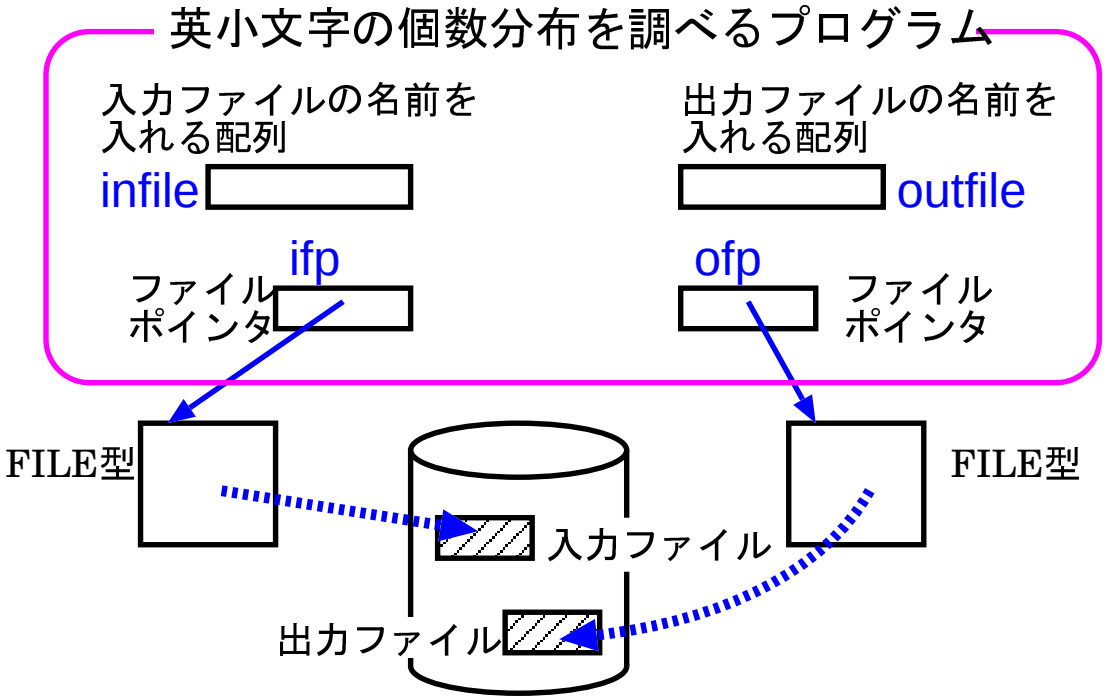
```
25    ave = 0.0;
26    for (i=0; i<50; ++i) {
27      fscanf(ifp, "%lf", &x[i]);
28      ave += x[i];
29    }
30    ave /= 50.0;

31    var = 0.0;
32    for (i=0; i<50; ++i)
33      var += (x[i]-ave)*(x[i]-ave);
34    var /= 50.0;

35    fprintf(ofp, "\nInput data:\n");
36    for (i=0; i<50; i+=5)
37      fprintf(ofp, "%14.5e%14.5e%14.5e%14.5e%14.5e\n",
38              x[i], x[i+1], x[i+2], x[i+3], x[i+4]);
39    fprintf(ofp, "\nAverage  = %14.6g\n"
```

```
    40              "Variance = %14.6g\n", ave, var);

    41    fclose(ifp);
    42    fclose(ofp);
    43    return 0;
    44 }
[motoki@x205a]$  gcc average-variance-io-through-files.c
[motoki@x205a]$  ./a.out
[motoki@x205a]$  cat out

Input data:
   1.00000e+00   1.00010e+00   1.00020e+00   1.00030e+00   1.0
   1.00050e+00   1.00060e+00   1.00070e+00   1.00080e+00   1.0
   1.00100e+00   1.00110e+00   1.00120e+00   1.00130e+00   1.0
   1.00150e+00   1.00160e+00   1.00170e+00   1.00180e+00   1.0
   1.00200e+00   1.00210e+00   1.00220e+00   1.00230e+00   1.0
   1.00250e+00   1.00260e+00   1.00270e+00   1.00280e+00   1.0
```

```
   1.00300e+00   1.00310e+00   1.00320e+00   1.00330e+00   1.0
   1.00350e+00   1.00360e+00   1.00370e+00   1.00380e+00   1.0
   1.00400e+00   1.00410e+00   1.00420e+00   1.00430e+00   1.0
   1.00450e+00   1.00460e+00   1.00470e+00   1.00480e+00   1.0

Average  =         1.00245
Variance =     2.0825e-06
[motoki@x205a]$
```

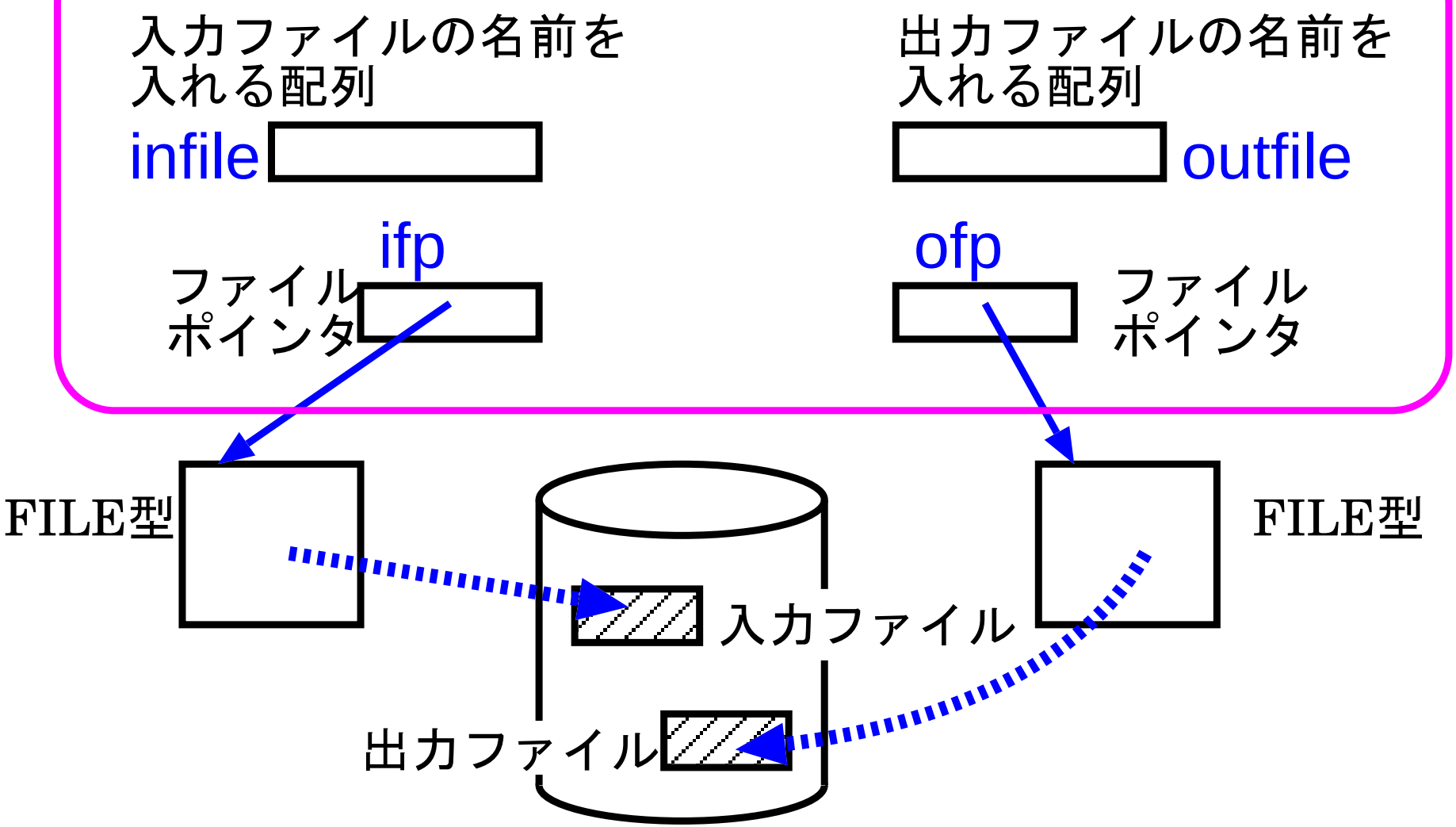
例題12. 5 (指定したファイル中の英小文字の個数分布)
①入力ファイル、出力ファイルの名前を標準入力から受け取り、
②入力ファイル中の英小文字の個数分布を出力ファイルに書き出す
Cプログラムを作成せよ。

(考え方)　入力ファイルと出力ファイルの名前を文字列として保持する十分長いchar型配列、それから入力ファイルに繋がるファイルポインタ, 出力ファイルに繋がるファイルポインタの領域が必要である。



拡大→次ページ

英小文字の個数分布を調べるプログラム



例題12. 5 (指定したファイル中の英小文字の個数分布)
①入力ファイル、出力ファイルの名前を標準入力から受け取り、
②入力ファイル中の英小文字の個数分布を出力ファイルに書き出す
Cプログラムを作成せよ。

(考え方)　入力ファイルと出力ファイルの名前を文字列として保持する十分長いchar型配列、それから入力ファイルに繋がるファイルポインタ, 出力ファイルに繋がるファイルポインタの領域が必要である。

これらのものが用意されていれば、

- ライブラリ関数 fopen() を利用して、char型配列で指定したファイルをオープンできる。 ($\Longrightarrow$ 10.8節を参照)
- ライブラリ関数 fclose() を利用して、ファイルポインタで指定したファイルをクローズできる。 ($\Longrightarrow$ 10.8節を参照)
- ライブラリ関数 fgetc() を用いて、ファイルポインタで指定した先から次のデータを1文字分だけ読み込むことができる。
- ライブラリ関数 fprintf() や fputc() を用いて、ファイルポインタで指定した先に出力データを流し込むことができる。

度数分布を調べるためには 、単に、英小文字26文字それぞれの出現回数を数えるカウンタ (初期値0) を用意して、入力ファイルから英小文字を検出する度にその文字のカウンタを1だけ増やす操作を続ければ良い。その際、

文字 'a' の出現回数を count_of_letter[0] で,
文字 'b' の出現回数を count_of_letter[1] で,
.....,
文字 'z' の出現回数を count_of_letter[25] で

数える場合は、

'a'-'a'=97−97=0,
'b'-'a'=98−97=1,
.....
'z'-'a'=112−97=25

⟹ 11.3節を参照

と計算できるので、一般には

変数 c に記憶されている文字(番号)の出現回数は
count_of_letter[c-'a'] で数える

ことになる。

(プログラミング)

ファイル名保持のためにinfile[]，outfile[] というchar型配列を、
ファイルポインタ保持のためにifp，ofp という変数を、そして
英小文字26文字の各々の出現回数を数えるために count_of_letter[]
というint型配列を
用意して、プログラムを構成した。

```
[motoki@x205a]$  nl counting-lowercase-letters.c
     1 /* (1)入力ファイル，出力ファイルの名前を標準入力から受け...
     2 /* (2)入力ファイル中の英小文字の個数分布を出力ファイルに...
     3 /* Cプログラム

     4 #include <stdio.h>
     5 #include <stdlib.h>
```

```
 6 int main(void)
 7 {
 8   int   c, i, count_of_letter[26];
 9   char  infile[100], outfile[100];
10   FILE  *ifp, *ofp;

11   printf("Type in a name of an input file:  ");
12   scanf("%99s", infile);
13   printf("Type in a name of an output file: ");
14   scanf("%99s", outfile);
15   if ((ifp=fopen(infile, "r")) == NULL) {
16     printf("ファイルをオープン出来ません: %s\n", infile);
17     exit(1);
18   }
19   if ((ofp=fopen(outfile, "w")) == NULL) {
20     printf("ファイルをオープン出来ません: %s\n", outfile)
21     exit(1);
22   }
```

```
23    for (i=0; i<26 ; ++i)        /* カウンタを全て0に初期化
24      count_of_letter[i] = 0;

25    while ((c=fgetc(ifp)) != EOF)
26      if (c>='a' && c<='z')
27        ++count_of_letter[c-'a'];

28    for (i=0; i<26; ++i) {
29      fprintf(ofp, "%c:%4d      ",
                  'a'+i, count_of_letter[i]);
30      if (i%5==4)
31        fputc('\n', ofp);
32    }
33    fputc('\n', ofp);
34    fclose(ifp);
35    fclose(ofp);
36    return 0;
```

```
    37 }
[motoki@x205a]$  gcc counting-lowercase-letters.c
[motoki@x205a]$  ./a.out
Type in a name of an input file:
                                  counting-lowercase-letters.c
Type in a name of an output file:  out
[motoki@x205a]$  cat out
a:  13      b:   1      c:  20      d:   6      e:  33
f:  47      g:   1      h:   4      i:  52      j:   0
k:   0      l:  20      m:   3      n:  34      o:  29
p:  23      q:   0      r:  15      s:  10      t:  34
u:  16      v:   1      w:   2      x:   2      y:   2
z:   1
[motoki@x205a]$
```

ファイルについてのまとめ：

- Cプログラムの中では(標準入出力も含めた)ファイルへのアクセスは、通常、ファイルポインタを介して行う。

- 内部的には、ファイルポインタは

  入力用か出力用か、
  次の読み込み文字の位置 (または次の書き込み場所)、
  ファイル終端が起きたかどうか、

  などの情報から成る(FILE型)構造体へのポインタであり、特に標準入力, 標準出力, 標準エラー出力にアクセスするためのファイルポインタとしては、<stdio.h>の中でそれぞれ stdin, stdout, stderr という名前のものが用意されている。

補足：

FILE型の定義はシステムによって異なっている様である。

例えば、平林雅英「ANSI C/C++辞典」(共立出版,1996)には FILE 型の定義として次の様なものが例示されている。

```
typedef struct {
  unsigned char *fpi;    /* ファイル位置指示子 */
  unsigned char *bptr;   /* バッファへのポインタ */
  unsigned int  flags;   /* ファイル状態フラグ */
    ..........................
```

- ファイル処理は一般に次の様に行う。

① `#include <stdio.h>`

② `FILE *` ファイルポインタ型変数 ;

③ ファイルポインタ型変数
`= fopen(` ファイル名`,` 使用モード `);`
但し、使用モード としては次の3つが可能。
- `"w"` ... 新規書き込み
- `"a"` ... 追加書き込み
- `"r"` ... 読み込み

④ (ファイル操作)

⑤ `fclose(` ファイルポインタ型変数 `);`