真珠婦人
ーPerlによるCGI入門ー

岐阜経済大学
経営学部 経営情報学科
井戸　伸彦

来歴：
0.0版　2002年11月5日

前提

- 本講座では、Perlを用いてインタラクティブなWebページであるCGIを作成する方法について説明します。
- 「ヘンタイ良い子のWeb講座」、「ツァラトストラ書くWeb　ー速習HTML入門ー」「シャボン玉HTML　ーフォーム入門ー」を受講している程度の知識がある受講者を想定しています。
- CGIを用いたインタラクティブなWebページを作るための、最小限の説明だけに留めています。必要な注意等も省略している場合があります。
- エディタについては、操作出来ることを前提としてます。

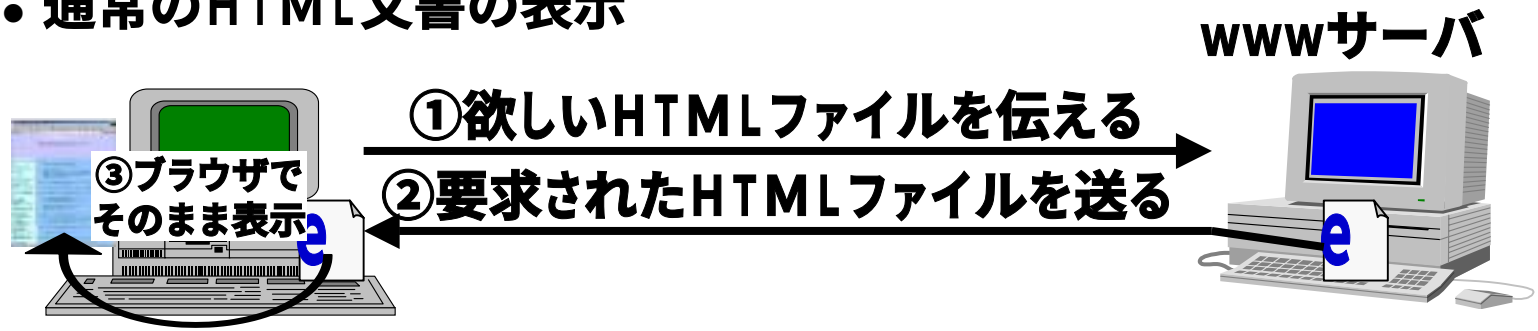
スライドの構成

- (1) CGIのしくみ
- (2) Unixサーバ上での操作
- (3) Perlプログラミング
- (4) 受信データを読み取る
- (5) 最初のプログラム
- (6) ファイルの操作
- (7) 制御文
- (8) ソフトウェアを上手に作る

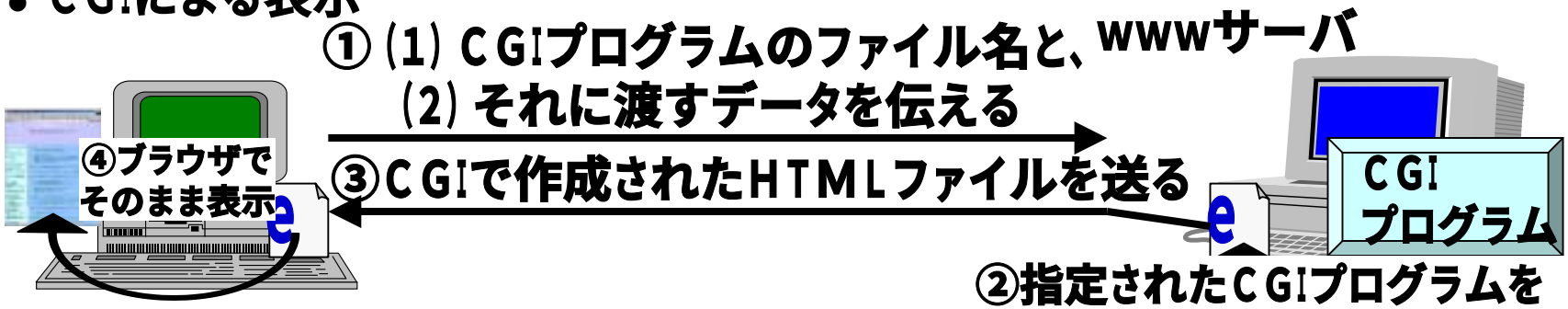
(1) CGIのしくみ

■CGIでは、HTML文書がサーバ側に用意されているのではなく、HTMLを作成するプログラムが用意されています。

- 通常のHTML文書の表示



- CGIによる表示



■CGIプログラム

- どのような言語で書いても良いが、普通はPerlで書く場合が多い。

(1．1) HTMLを作るプログラム

- 難しく考える必要はありません。HTMLをプリント文で書き出せば良いだけです。

＜HTML＞

```
1. <html>
2. <head>
3. <title>クラス掲示板</title>
4. </head>
5. <body>
6. hello, world.
7. </body>
8. </html>
```

＜PerlによるCGI：clsBoard.cgi＞

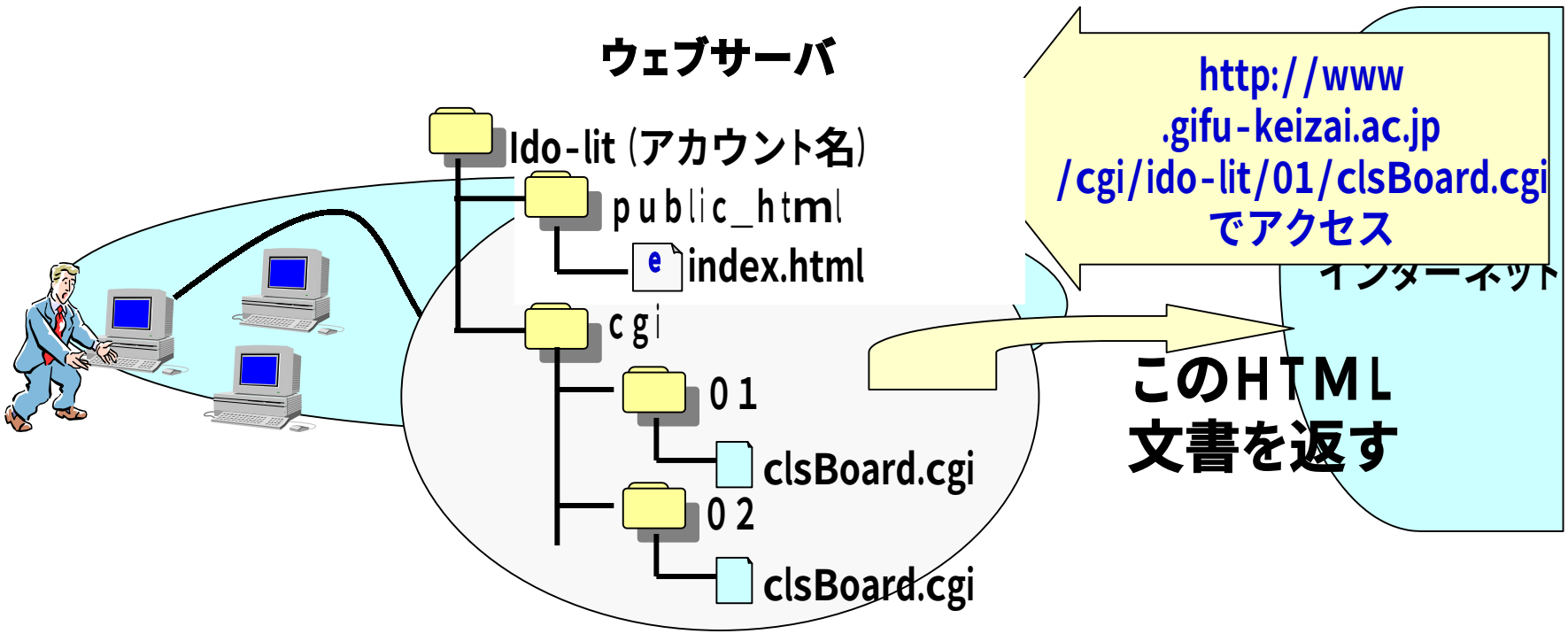
```
1. #!/usr/local/bin/perl
2. print "Content-type:
   text/html¥n¥n";
3. print "<html>¥n";
4. print "<head>¥n";
5. print "<title>クラス掲示板
   </title>¥n";
6. print "</head>¥n";
7. print "<body>¥n";
8. print "hello, world.¥n";
9. print "</body>¥n";
10.print "</html>¥n";
```

- CGIを作成するファイルの名前には、末尾に拡張子の“.cgi”を付けます。
- ファイルの先頭には、“#!/usr/local/bin/perl”を入れます。

(1．2) CGIプログラムの置き場所

■作成したファイルは、cgiの実行が許された特定のアカウント上の、特定の位置に置くことで、外部から参照可能となります。

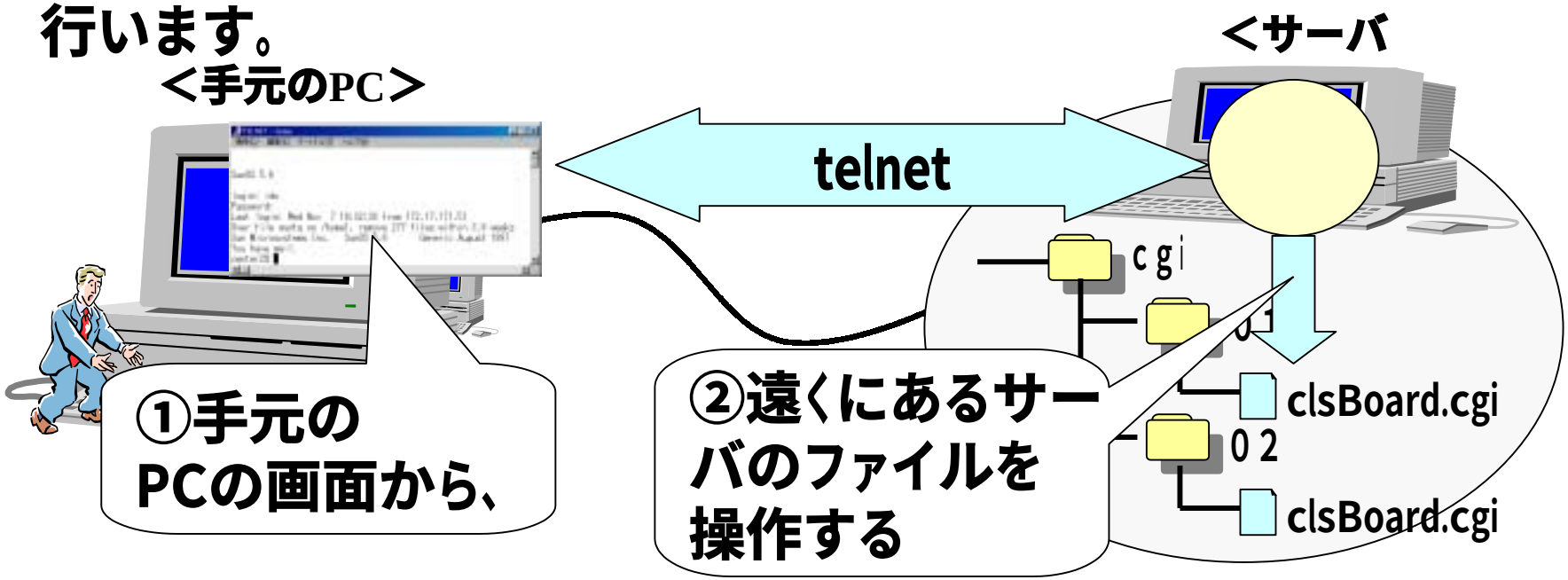
■今回は、“ido-lit”のアカウントを用います。



(2) Unixサーバ上での操作

■CGIのプログラムは、Unixのサーバ上で実行されます。このため、Unix上での操作が必要になります。

■telnetと呼ぶ方法を使って、手元のPCから、サーバ上の操作を行います。



■操作は、アイコンやマウスを使って行う方法でなく、コマンドラインからコマンドを打ち込むことで行います。どんなものか、実際に試しながら覚えていきます。

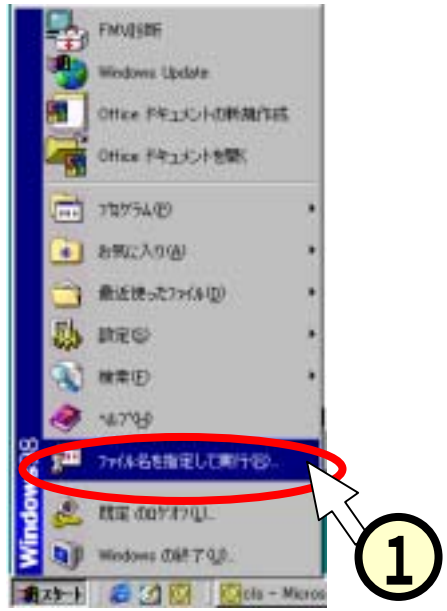
(2．1) Unixのコマンドラインの起動

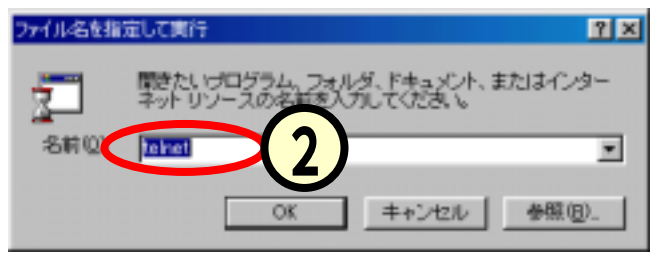
■Telnet

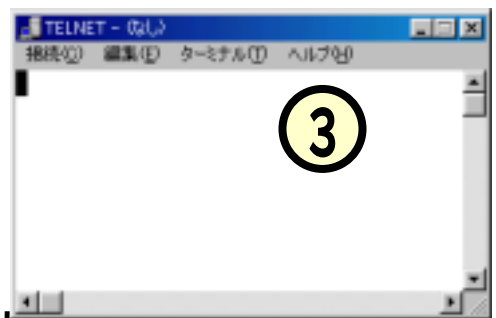
- 遠隔のコンピュータに入るための手段(ソフトウェアとプロトコル)です。

■手順

- [スタート]-[ファイル名を指定して実行(R)](①)を選択する。
- 出てきたウィンドウにて、“telnet”と入力(②)する。
- ③のようなウィンドウが現れます。







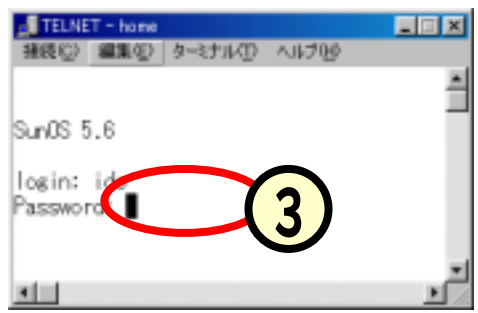
(2．2) アカウントにログインする

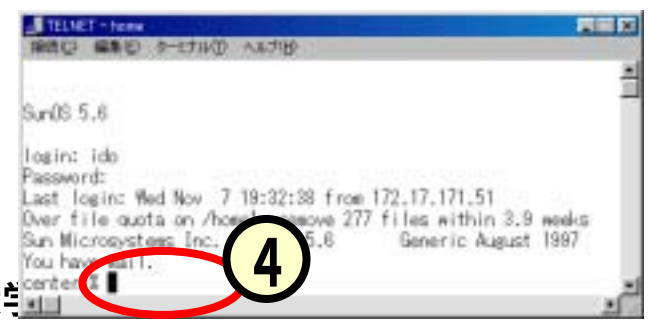
■ログイン

- 計算機の使用を開始することと思ってください。

■手順

- [接続]-[1 home]を選択 (①) します。
- 画面上で、アカウント名の入力が促されますので、これに自分のアカウント (cxxxxxxx) を入力 (②) します。
- 同様に、パスワードの入力が促されますので、これに自分のパスワードを入力 (③) します。ただし、画面上で打ち込んだパスワードは表示されません。
- ログインが完了し、コマンドライン (コマンドを打ち込む行) が出ます (④)。



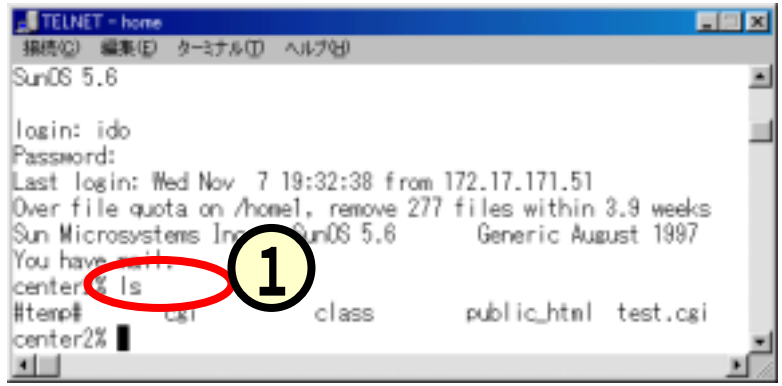


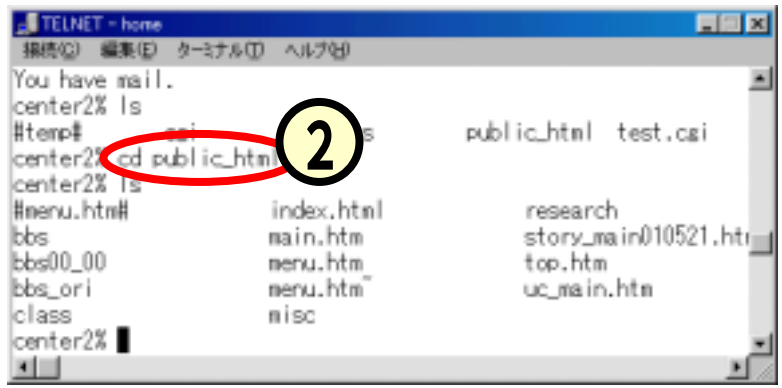
(2．3) ファイルを確かめる。

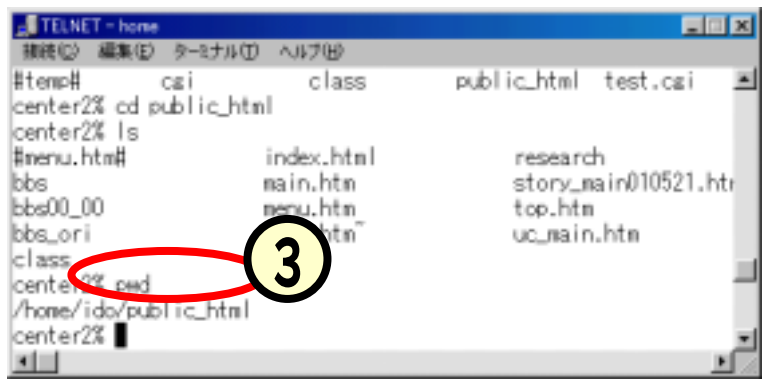
■コマンド

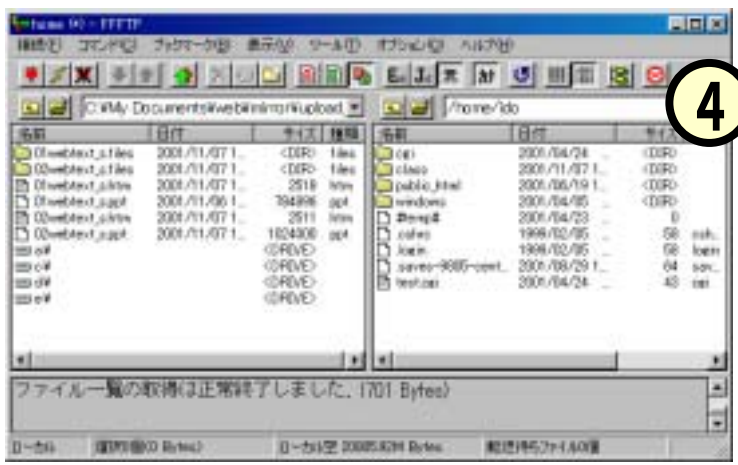
- ls : ファイル一覧、
  ディレクトリ一覧を表示 (①)。
- cd xxx : ディレクトリxxxへ移動 (②)。
- cd .. : ひとつ上のディレクトリへ
  移動。
- pwd : 今どこのディレクトリにいるかを
  表示 (③)。

■FTPでのファイルの見え方 (④)
と比べてみてください。









(2．4) パーミッション

■CGIを使うための操作をここで説明します。

■Unix上では、ファイルはモードと呼ばれるものを持っています。
次のように“ls -l”でこれを見ることが出来ます。

- % ls –l
- drwxr-xr-x 2 ido-lit ftea 512 11月 7日 10:47 09
- -rwxr-xr-x 1 ido-lit ftea 2919 11月 6日 14:37 classBoard.cgi
- -rw-r--r-- 1 ido-lit ftea 91 11月 6日 14:37 classboard.txt
- -rwxr--r-- 1 ido-lit ftea 404 11月 7日 11:38 clsBoard.cgi

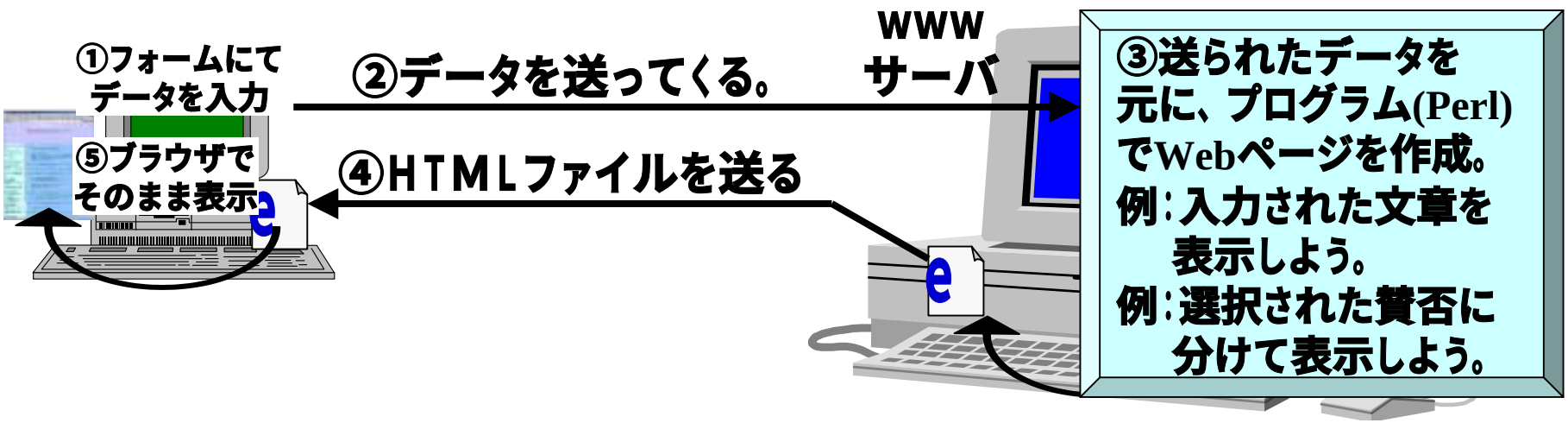
一般ユーザが、r：読み出すが出来る
w：書き込むことが出来る
x：プログラムとして実行することが出来る

■スライド(1．1)で作成したCGIファイルをFTPで転送し、次のようにモードを変更して、Webブラウザで見てみましょう。

- chmod a+x clsBoard.cgi

(3) Perlプログラミング

- スライド(1．1)のように、HTMLを書き出すだけならば、CGIを使う必要はありません。
- 「シャボン玉HTML－フォーム入門－」で学んだ、フォームを使ったWebサイトから送られてくるデータをプログラムで処理することにより、入力に応じたWebページを作っていくのがCGIの目的です。
- 使用するプログラミング言語が、“Perl”という訳です。



(4) 受信データを読み取る

```
#!/usr/local/bin/perl
require 'jcode.pl';
&get_parameter;
<<<<ここにプログラムを書いていく>>>>
sub get_parameter {
        print DBG "#D#:get_parameter:begining¥n";
        if ($ENV{'REQUEST_METHOD'} eq "POST") {
                read(STDIN, $QUERY_DATA, $ENV{'CONTENT_LENGTH'});
        } else { $QUERY_DATA = $ENV{'QUERY_STRING'}; }
        @pairs = split(/&/,$QUERY_DATA);
        foreach $pair (@pairs) {
                ($name, $value) = split(/=/, $pair);
                $value =~ tr/+/ /;
                $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
                $value =~ s/¥n//g;
                $value =~ s/¥,/, /g;
                &jcode'convert(*value,'sjis');
                $value =~ s/</&lt;/g;
                $value =~ s/>/&gt;/g;
                if ($name ne 'comment') {
                        $value =~ s/ /　/g;
                        $value =~ s/¥!/！/g;
                }
                $FORM{$name} = $value;
                print DBG "#D#:get_parameter:010:name=".$name.",value=".$value."¥n";
        }
}
```

データの読み取りは、後回しにします。ひとまず、このプログラムをコピーしておいてください。

(5) 最初のプログラム

■次のようなプログラムを作って動かしてみます。前スライドの赤い字の部分に入れる訳ですね。

```
 1 print "Content-type: text/html¥n¥n";
 2 print "<!DOCTYPE HTML PUBLIC -//IETF//DTD HTML//EN>¥n";
 3 print "<html>¥n";
 4 print "<head>¥n";
 5 print "<meta http-equiv=Content-Type content=text/html; charset=x-sjis>¥n";
 6 print "<title>クラス掲示板</title></head>¥n";
 7 print "<body>¥n";
 8 print "<form action=clsBoard.cgi method=POST>¥n";
 9 print "hello,".".$FORM{yourName}"."<br>¥n";
10 print "<input type=¥"text¥" name=¥"yourName¥" ><br>¥n";
11 print "<input type=¥"submit¥" value=¥"送信¥">¥n";
12 print "</form>¥n";
13 print "</body>¥n";
14 print "</html>¥n";
```

■赤線の部分は、おまじないだと思っておいてください。

■赤線の部分以外の部分について、説明していきます。

(5.1)print文

■形式
- print <ファイルハンドル> <文字列のリスト>;

文末には、セミコロン“;”が必要。

■print "abcd";
- 「abcd」という文字列をプリントする。

文字列は、“”で囲う

■print "abcd¥n";
- 「abcd」という文字列をプリントして、改行する。
- Unixでは、¥nは、バックスペース(“\”)。

■print "abcd"."efg¥n";
- 「abcdefg」という文字列をプリントして、改行する。

「 . 」は、2つの文字列を連結する。

■$var="hij";
```
print "abcd".$var."¥n";
```
- 「abcdhij」という文字列をプリントして、改行する。
- $varは変数。次のスライドを参照。

(5．2)「"」をプリントする

■次のように書くと、エラーになります。

✕ `print """;`

文字列の始まりを示す、「"」①

プリントしたい文字である、「"」②

文字列の終わりを示す、「"」③

→ コンピュータには、②と③との区別がつかない。

■文字列(＝"xxx")中に、「"」を入れる場合は、「¥"」と書きます。「¥」のことを、エスケープ文字と呼びます。

○ `print "¥"";`

エスケープ文字「¥」の後の「"」は、そのままプリントされる。

■HTMLでは、「"」を良く使うので、CGIでは、「¥"」を良く使います(Unixでは、「¥」は、「\」(バック・スラッシュ)になります)。

(5．3) 変数

■Perlには、3種類の変数があります。

- スカラ変数：「$」で始まる、1つだけの値を記憶する変数。
  - Perlでは、変数の宣言は特に必要ない。他の変数も同じ。
  - 数値でも、文字列でも、代入が出来る。

```
1. $a=50;
2. $b=65;
3. $c=$a+$b;
```
「$cには、“115”が入る」

```
1. $a=“aa”;
2. $b=“bb”;
3. $c=$a.$b;
```
「$cには、“aabb”が入る」

「．」は、2つの文字列を連結する。

- 配列変数：「@」で始まる、複数の値を記憶する変数。
  - 「@arr」を参照/代入するときは、「$arr[引数]」のような形で書く。

```
1. $a[0]=50;
2. $a[1]=65;
3. $a[2]=$a[0]+$a[1];
```
「$a[2]には、“115”が入る」

```
1. $a[0]=“aa”;
2. $a[1]=“bb”;
3. $a[2]=$a.$b;
```
「$a[2]には、“aabb”が入る」

(5．4) 連想配列

■連想配列：

- 「%」で始まる、複数の値を記憶する変数。
- 「%hash」を参照/代入するときは、「$hash{引数}」のような形で書く。
- 引数に文字列が使える。←これが便利！

```
1. $a{“apple”}=50;
2. $a{“orange”}=65;
3. $a{“sum”}=$a{“apple”}+$a{“orange”};
```

「$a{“sum”}には、“115”が入る」

```
1. $a{“apple”}=“red”;
2. $a{“orange”}=“orange”;
3. $a{“color”}=$a{“apple”}.$a{“orange”};
```

「$a{“sum”}には、“appleorange”が入る」

(5．5) 我々のプログラムでは。。

■9行目に、連想配列があります。

```
9 print "hello,".$FORM{yourName}."<br>¥n";
```

- 「%FORM」という連想配列の、「yourName」で索引される値を参照してます。これで、“hello,”の後に名前が入ります。

■誰が値を設定したか？

- 実は、スライド (4) のプログラムが、値を入れています。

<ブラウザ>

①ここに“ido”と入れて、[送信]をクリックすると、

②このようなデータが送られ、

yourName=ido

③スライド (4) のプログラムにて、「$FORM{yourName}=“ido”」を設定される。

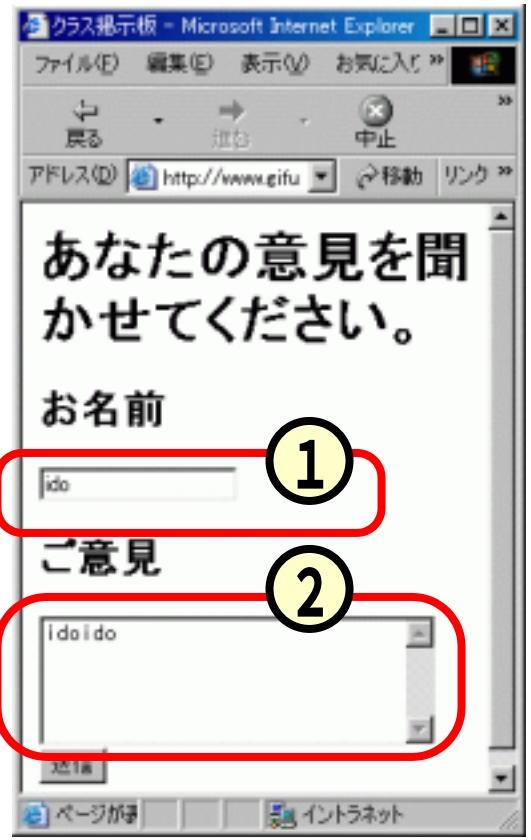
```
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-
    $value =~ s/¥n//g;
    $value =~ s/¥,/, /g;
    &jcode'convert(*value,'sjis');
    $value =~ s/</&lt;/g;
    $value =~ s/>/&gt;/g;
    if ($name ne 'comment') {
        $value =~ s/ /  /g;
        $value =~ s/¥!/ ! /g
    }
    $FORM{$name} = $value;
    print DBG "#D#:get_parameter:010:name=",$name,",value=",$value."¥n";
}
```

$FORM{$name} = $value;

(5．6) 問題1

■右図のようなCGIプログラムを作成してください。

- ①は、テキスト･フィールドです
- (シャボン玉 (2.4) 参照)。
- ②は、テキスト･エリアです (同 (3.1) 参照)。
- ①、②のように入力して送信ボタンを押すと、③、④のように変わるようにします。



■ヒント

- テキスト･フィールドでは、“value”属性を、
- テキスト･エリアでは、初期値の部分に設定します。
- Webページを開いた最初から、「氏」、「と思う。」は表示されていて、OKです。

(5．7)解答1

■解答例

```
1. print "<body>¥n";
2. print "<form action=clsBoard.cgi method=POST>¥n";
3. print "<h1>あなたの意見を聞かせてください。</h1>¥n";
4. print "<h2>お名前</h2>¥n";
5. print "<input type=¥"text¥", name=¥"yourName¥"
   value=".$FORM{yourName}."氏".">¥n";
6. print "<h2>ご意見</h2>¥n";
7. print “<textarea name=¥”yourComment¥“ rows=5
   cols=30>¥n”.$FORM{yourComment}.“と思う。
   "."¥n</textarea><br>¥n";
8. print "<input type=¥"submit¥" value=¥"送信¥">¥n";
9. print "</form>¥n";
10.print "</body>¥n";
```

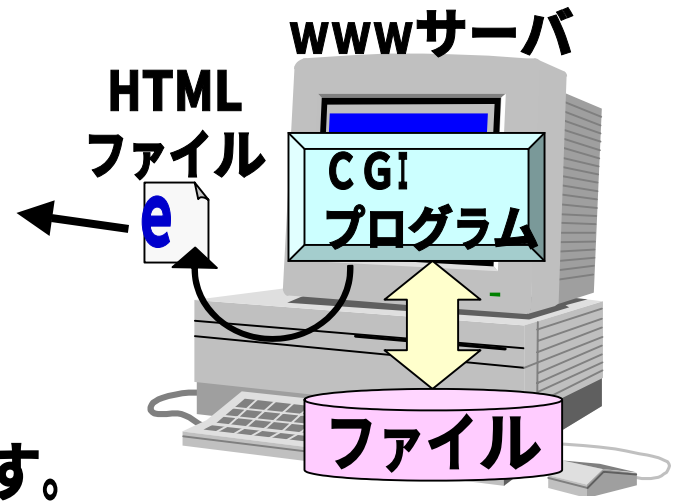
※：5行目、7行目は、改行しないで続けて書くようにお願いします。

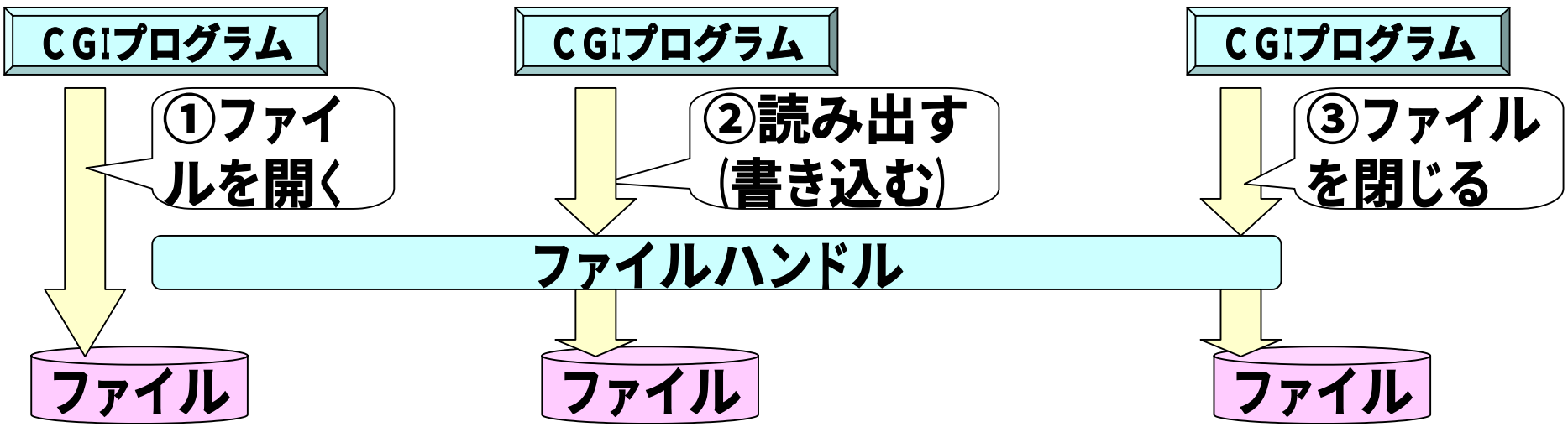
(6) ファイルの操作

■送信されたデータだけでは、
限られた内容のHTMLファイル
しか作成できません。

■ファイルを操作することで、
データを蓄積しておくことにします。

■ファイルを操作する手順は、次の通りです。





ファイルを開く際に、“ファイルハンドル”を指定し、これに対して
読み書きを行って、最後にファイルハンドル (=ファイル) を閉じる。

(6．1) テキストファイル、構成

■ここで扱うファイルは、テキストファイルです。すなわち、メモ帳などのテキストエディタで扱うファイルです。

■テキスト構成

- データをテキストファイルに蓄積する場合、いろいろな方法が考えられます。この方法を、テキスト構成と呼ぶことにします。

＜ファイル“opinion”のテキスト構成＞

例1

```
井戸、賛成です
草薙、反対です
香取、反対です
```

例2

```
(1)井戸
(2)草薙
(3)香取
[1]賛成です
[2]反対です
[3]反対です
```

例3

```
井戸
賛成です
草薙
反対です
香取
反対です
```

- テキストファイルの場合、行をデータの単位とするテキスト構成がほとんどです。
- さらに、行を区切り記号 (上記の例1では、“、”) で、いくつかのフィールド (部分) に分けて使うやり方は、一般的です。

(6．2)配列にファイルを読み出す

■前スライド(6．1)の例1のファイル“opinion”を開いて配列に収め、各行の内容を打ち出すプログラムは、次のようになります。

①ファイルを開く(open)

ファイルハンドル名は、“DB”
(各自命名すれば良い)

②ファイルを読み出す
(配列“@data”に収める)。

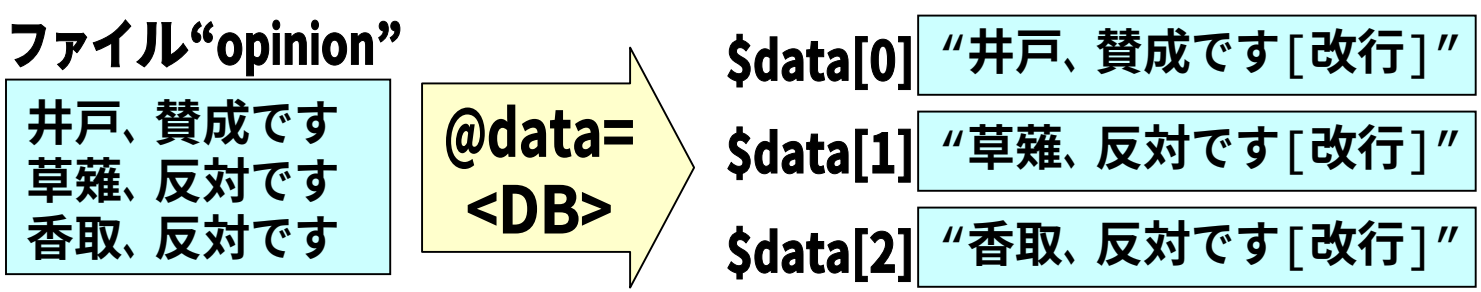
③ファイルを閉じる

```
open(DB,“opinion");
@data=<DB>;
close(DB);
print $data[0];
print $data[1];
print $data[2];
```

開くファイルのファイル名は、“opinion”

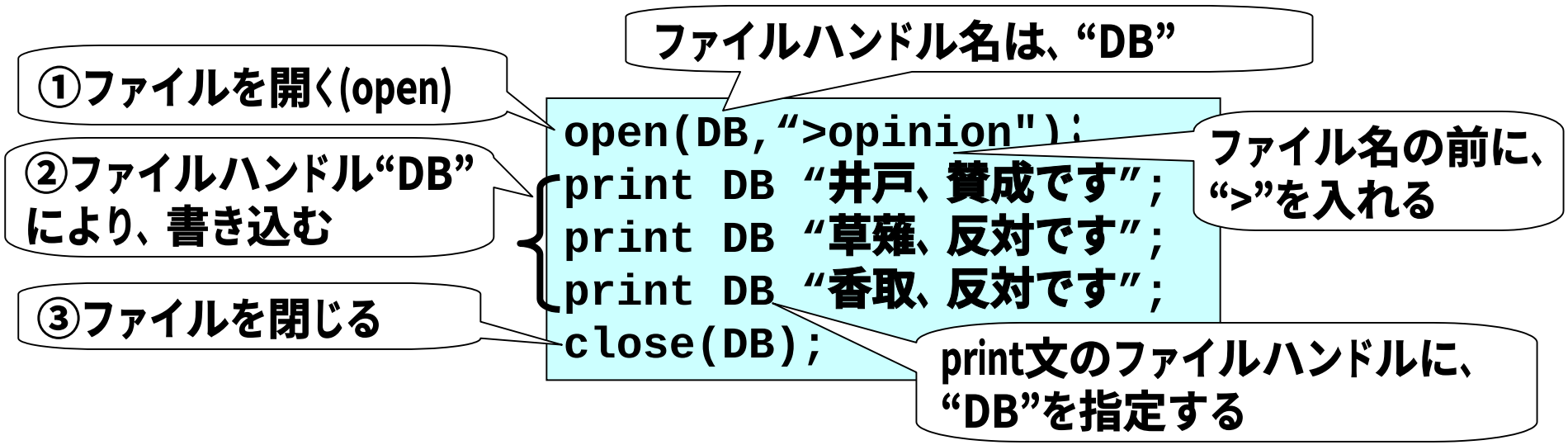
“<>”でファイルハンドルを囲う

■ファイルハンドルの配列への代入は、次のように、行が各配列要素になるように行われます。



(6．3) ファイルに書き出す

■スライド (6．1) の例1のファイル“opinion”を作成するプログラムは、次のようになります。



■追加書き込み (ファイルの前の内容を消さず、末尾に追加する) 用にファイルを開くときは、“>>”を使います。

- Open(DB,”>>opinion”);

(6．4) フィールドを分割する

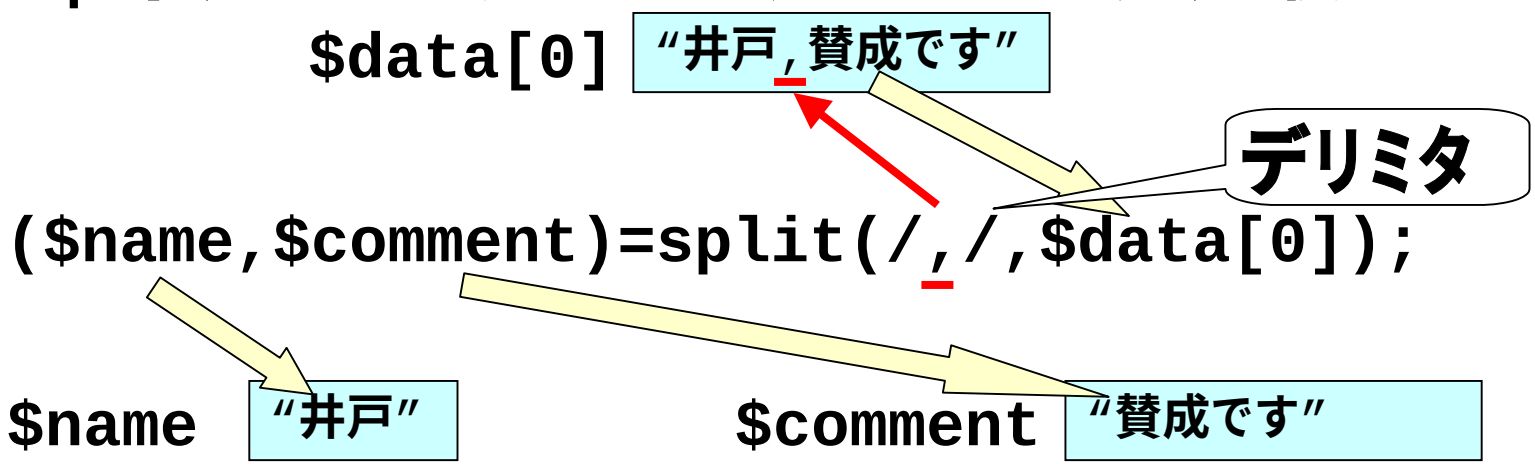
■目的：$data[0]の中の「井戸,賛成です[改行]」を「井戸」と「賛成です」に分割する。

■chop：文字列の最後の[改行](改行記号)を取り除く。

- 「chop」(”空手チョップ“のチョップ)を使う。

■split：文字列をデリミタ(区切り文字)にて分割する。

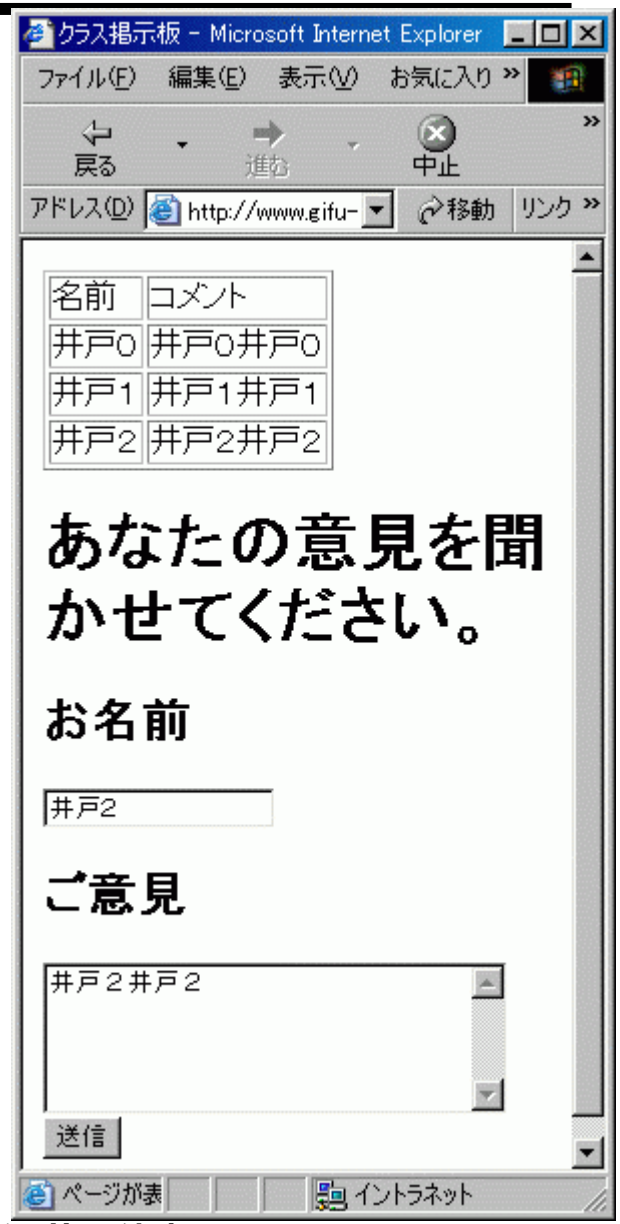
- 「split」(”ボーリングのスプリット“のスプリット)を使う。



(6．5)問題2

■過去3回分の投稿を表示する掲示板を作成せよ。

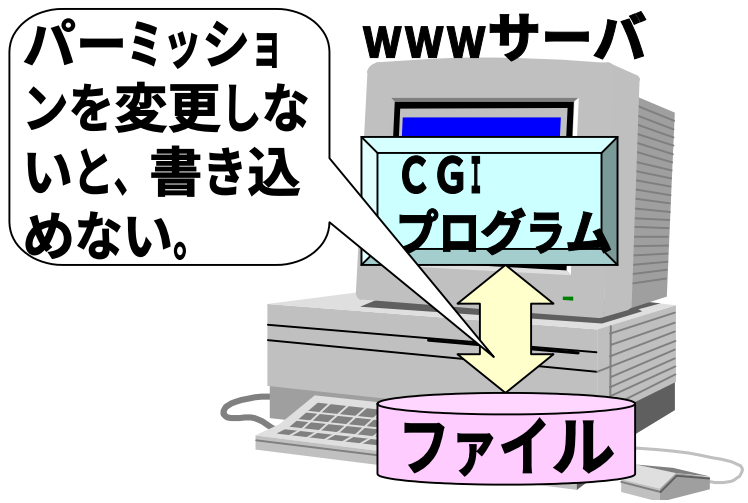
- 右のページは、次のような送信を行った後の画面です。
  - お名前=“井戸0”、ご意見=“井戸0井戸0”
  - お名前=“井戸1”、ご意見=“井戸1井戸1”
  - お名前=“井戸2”、ご意見=“井戸2井戸2”
- データを蓄積しておくファイルについては、次のスライドに従って、パーミッションを変更しておく必要があります。
- スライド(5．6)問題1の「氏」、「と思う。」を付加する機能は、削除してください。



(6．6) データファイルのパーミッション

■スライド(2．4)と同様に、
ファイルのパーミッションを
変更します。

■CGIプログラムがファイルに
書き込むことが
出来るようにする訳です。



■データファイルの名前を、“clsBoard.txt”とすると、次のような操作をUnix上で行います。

- `touch clsBoard.txt` # ファイルを作成
- `chmod a+w clsBoard.txt` # パーミッションを変更

すべてのユーザ(=a)に、書き込む(=w)権利を追加(=+)する。

（6．7）解答2（解答例：<body>の後に挿入）

```
1.  $datafile="clsBoard.txt";
2.  open(DB,"$datafile");
3.  @data=<DB>;
4.  close(DB);
5.  print "<table border=1>¥n";
6.  print "<tr><td>名前</td><td>コメント</td></tr>¥n";
7.  chop($data[0]);
8.  ($namae,$iken)=split(/,/,$data[0]);
9.  print "<tr><td>".$namae."</td><td>".$iken."</td></tr>¥n";
10. chop($data[1]);
11. ($namae,$iken)=split(/,/,$data[1]);
12. print "<tr><td>".$namae."</td><td>".$iken."</td></tr>¥n";
13. print
    "<tr><td>".$FORM{yourName}."</td><td>".$FORM{yourComment}."
    </td></tr>¥n";
14. print "</table>¥n";
15. open(DB,">$datafile");
16. print DB $data[1]."¥n";
17. print DB $FORM{yourName}.",".$FORM{yourComment}."¥n";
18. close(DB);
```

(7) 制御文

■制御文は、どうして必要か？

■分岐

- 例えば、右上図の処理では、1行目の処理で、ファイルを開くことに失敗すれば、2～6行の処理は行うべきではありません。
- 2～6行を行うか否かの分岐が必要になります。

■繰り返し

- 例えば、右下図の処理では、1～3行と、4～6行とでは、引数([0],[1])以外は同じ処理です。
- 100回分の投稿を表示することにしたら、同じことを100回書くことになり、無駄です。繰り返しを指定する方法が必要になります。

```
1. open(DB,“opinion");
2. @data=<DB>;
3. close(DB);
4. print $data[0];
5. print $data[1];
6. print $data[2];
```

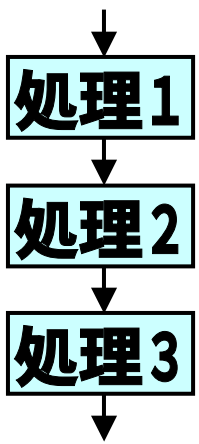
```
1. chop($data[0]);
2. ($namae,$iken)=split(/,/,
   $data[0]);
3. print
   "<tr><td>".$namae."</td><
   td>".$iken."</td></tr>¥n";
4. chop($data[1]);
5. ($namae,$iken)=split(/,/,
   $data[1]);
6. print
   "<tr><td>".$namae."</td><
   td>".$iken."</td></tr>¥n";
```

(7．1) プログラムの構造

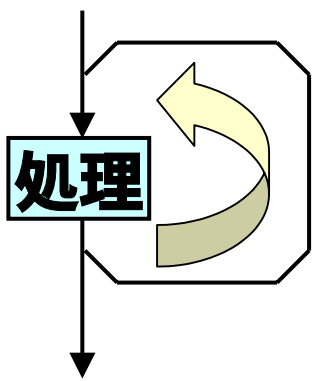
■3つの基本要素

- 「プログラムは複雑なもの」と考えておられる方もいると思いますが、視点を変えてみると、案外単純とも考えることが出来ます。
- 制御構造について言うと、次の3つを組み合わせることによって、プログラムは構成されています。

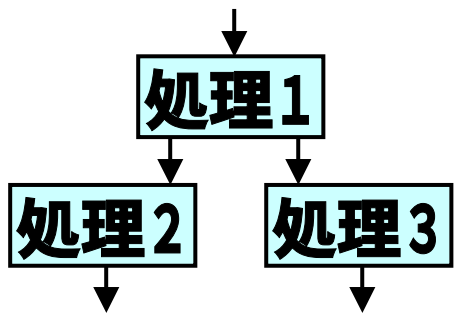
①連接：
順次実行する



②反復：
繰り返し実行する



③選択：
2つ以上の道筋に分岐する。

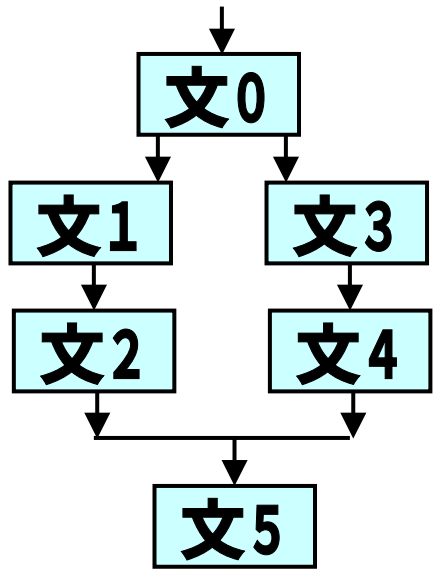


(7．2) if文

■形式
- 右図のとおり。

■制御式
- 右図にて、文1・文2に進むか、文3・文4に進むかは、制御文にて決まります。

```
1. 文0;
2. if(制御式){
3.   文1;
4.   文2;
5. }else{
6.   文3;
7.   文4;
8. }
9. 文5;
```



■簡単な例
- aが0の場合は、2行目により、b=1となる。
- aが0の場合以外は、4行目により、b=2となる。

```
1. if ($a==0){
2.   $b=1;
3. }else{
4.   $b=2;
5. }
```

(7．3) 制御式

■比較演算子

- 数値の比較 ： ($a=5;$b=7;)
  - $a != $b ⇒ 真 (成り立つ)
  - $a >= $b ⇒ 偽 (成り立たない)
- 文字列の比較 ：

($a="abc";$b="cde";$c=abc)

  - $a eq $c ⇒ 真 (成り立つ)
  - $a ge $b ⇒ 偽 (成り立たない)

比較	数値	文字列
等しい	==	eq
等しくない	!=	ne
より小さい	<	lt
より大きい	>	gt
より小さいか等しい	<=	le
より大きいか等しい	>=	ge

■関数の戻り値 (例：ファイルを開く) ： $r=open(DB,"file");

- 結果OK ⇒ $rは"真" 、 結果NG ⇒ $rは"偽"
- よって、ファイルがうまく開いた時のみ処理を行いたいのであれば、右図のような処理を行う。
  - ファイルがうまく開いた場合のみ、行3～6が実行される。
- 行1，2は、次のようにも書ける。

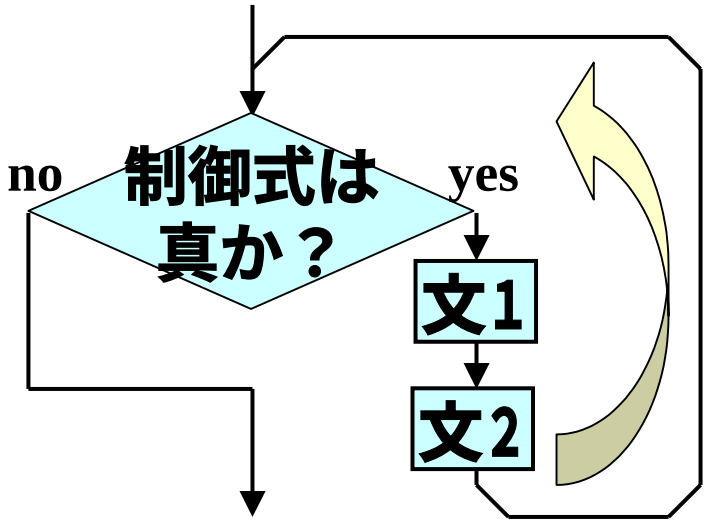
```
1. if(open(DB,"opinion")){
```

```
1. $r=open(DB,"opinion");
2. if($r){
3.    @data=<DB>;
4.    close(DB);
5.    print $data[0];
6.    print $data[1];
7. }
```

(7.4)while文

■形式 ： 図のとおり。

```
1. while(制御式){
2.   文1;
3.   文2;
4. }
```



■簡単な例

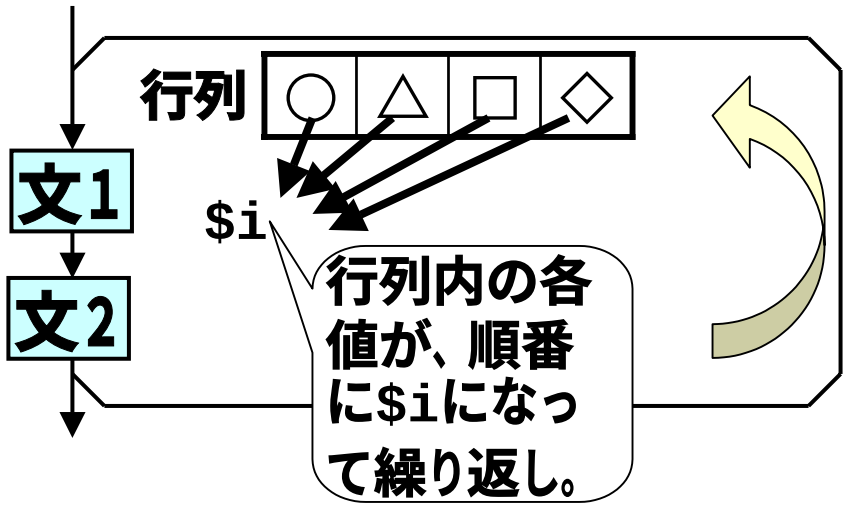
- 右のプログラムの動き
  - ◆行1：$a=3 $b=0
  - ◆行2：$a=3 $b=0
  - ◆行3：$a=3 $b=2
  - ◆行4：$a=2 $b=2
  - ◆行2：$a=2 $b=2
  - ◆行3：$a=2 $b=4
  - ◆行4：$a=1 $b=4
  - ◆行2：$a=1 $b=4
  - ◆行3：$a=1 $b=6
  - ◆行4：$a=0 $b=6
  - ◆行2：$a=0 $b=6
  - ◆行5：$a=0 $b=6
  - ⇒終了

```
1. $a=3;$b=0;
2. while($a>0){
3.   $b=$b+2;
4.   $a=$a-1;
5. }
```

(7．5) foreach文

■形式 ： 図のとおり。

```
1. foreach $i(行列){
2.  文1;
3.  文2;
4. }
```



■簡単な例

- 右のプログラムの動き
  - 行1：略
  - 行2：　　$s=“”
  - 行3：$i=“r”　$s=“”
  - 行4：$i=“r”　$s=“r”
  - 行3：$i=“g”　$s=“r”
  - 行4：$i=“g”　$s=“rg”
  - 行3：$i=“b”　$s=“rg”
  - 行4：$i=“b”　$s=“rgb”
  - 行5：　　⇒終了

```
1. @ar=(“r”,”g”,”b”)
2. $s=“”;
3. foreach $i (@ar){
4.  $s=$s.$i;
5. }
```

「．」は、2つの文字列を連結する。

（7．6）問題3

■右図のような掲示板を作ってください。

■動作

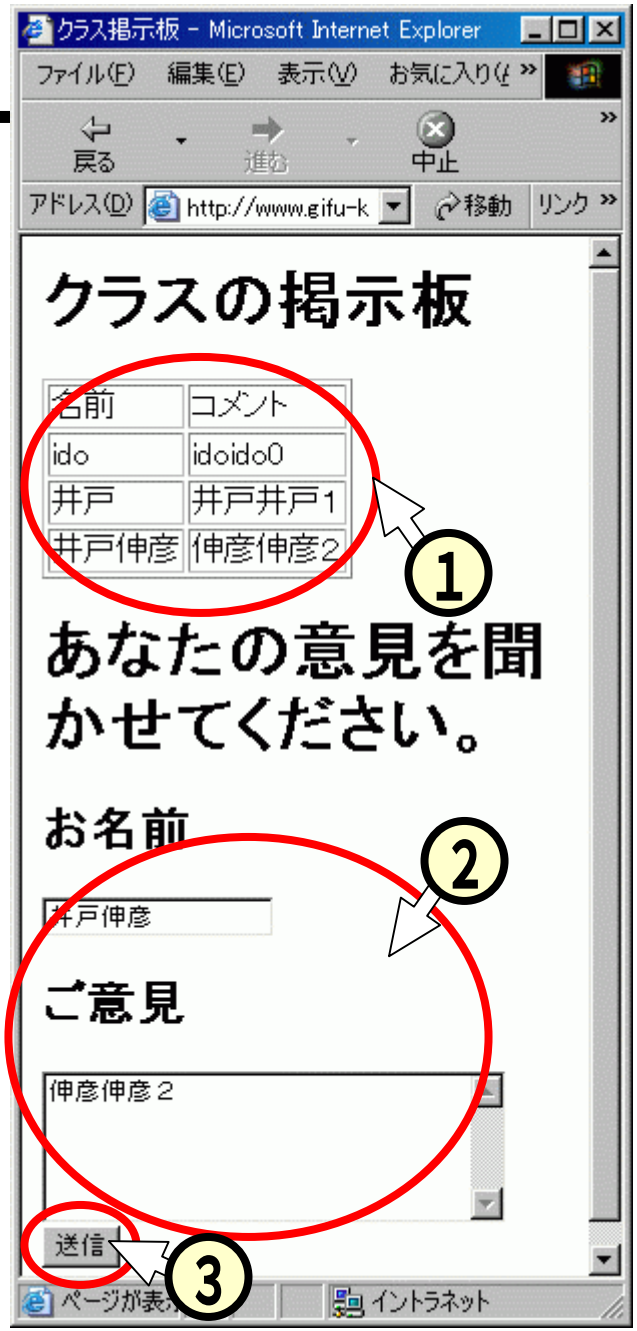
- ページにアクセスすると、各メンバの名前と意見の一覧が、①に表示されます。
- 自分の意見を書き込む際には、②に名前と意見を書き込んで、③のsubmitボタンをクリックします。
- 意見を書き込むと、上書きされる。前の意見を消すときは、名前だけ記入してsubmitボタンをクリックします。

■URL

- http://www.gifu-keizai.ac.jp/cgi/ido-lit/ido/clsBoard.cgi

■解答例

- /home/ido-lit/cgi/ido/clsBoard.cgi参照。



(8) ソフトウェアを上手に作る

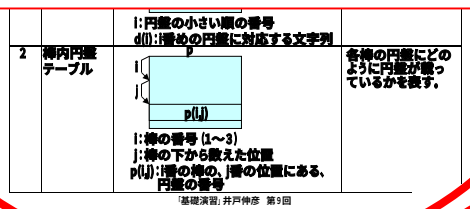
■ソフトウェア開発に携わった経験のある者には、「ソフトウェアを上手に作る」ということに、言葉には尽くせない思いがあります。この議論には、稿を改めて取り組みたいと思います。

■ここでは、指しあたって初心者の皆さんに考えて頂きたいことを記していきます。

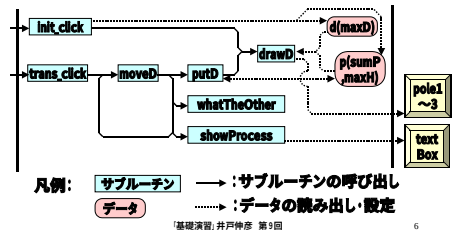
(8．1) プログラム設計の要素

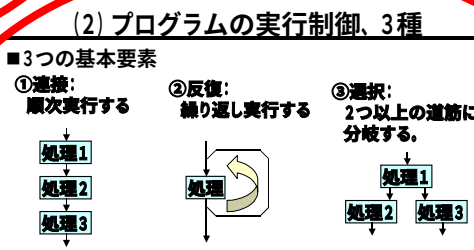
■プログラム設計には、次の3つの要素があります。

①データ構成：
目的に適った合理的な
データ構成を考える



②プログラム構成：
適切なサイズと互いの
関係を持ったプログラムに分割する





③処理手順：
平明なプログラムを書く

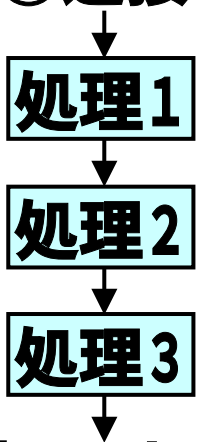
■上記の3つは、互いに関係します。

■本稿では、③処理手順について考えます。

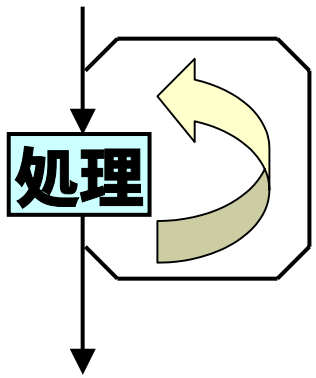
(8．2) 構造化プログラム

■前述のとおり、次の3つを組み合わせてプログラムは作られます。

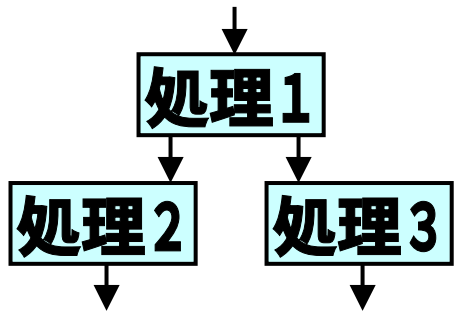
①連接：



②反復：



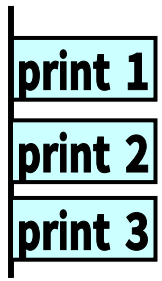
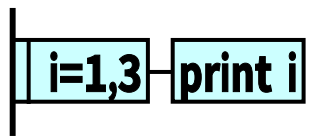
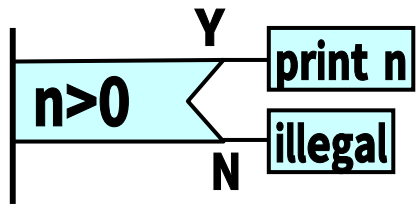
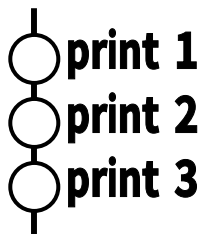
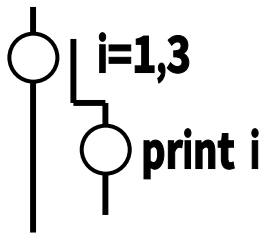
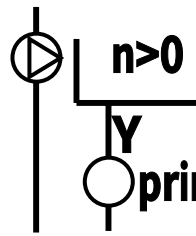
③選択：



■手続き型であれば、プログラミング言語によらず、この3つをどのように組み合わせるかにより、プログラムの良し悪しが決まります（言語に依存する部分も実際にはありますが）。

■次のスライドにて、組み合わせ方を記す記法を紹介します。

(8．3) さまざまな記法

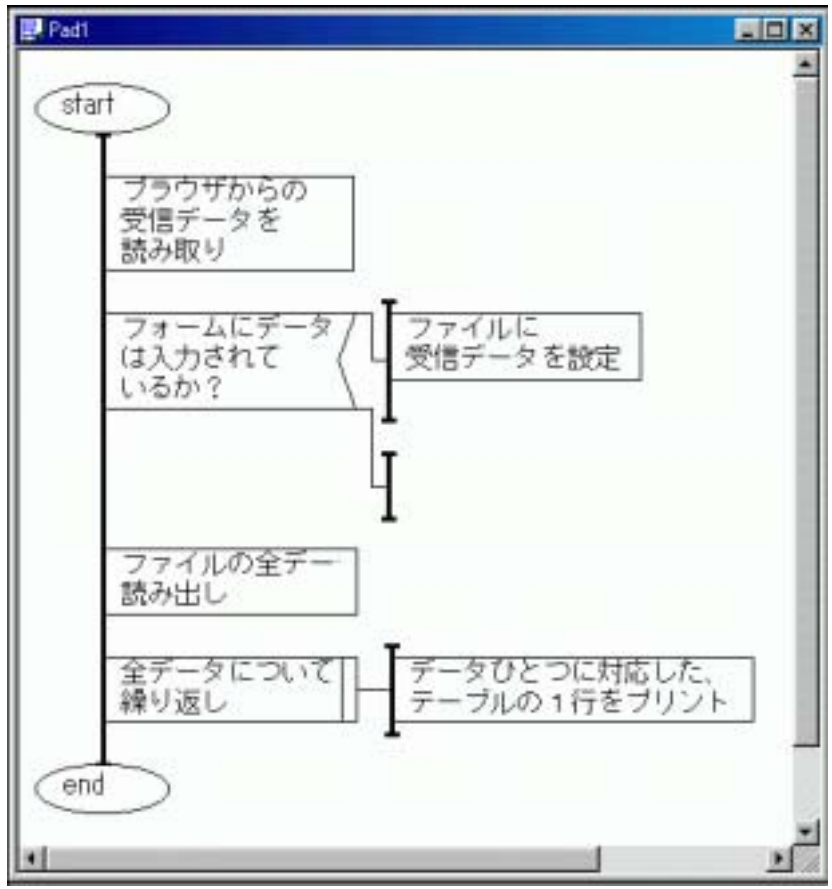
	①連接：	②反復：	③選択：
PAD			
HCP チャート			
IPC	： 2.1 print 1 2.2 print 2 2.3 print 3 同じネスト(＝書き始めの位置と番号の桁)の続きの番号は、順次実行。	： 2.1 i=1,3にて繰り返し 2.1.1 print i 繰り返しの部分は、ネストをひとつ下げる。	2.1 n>0かを判定 2.1.1 n>0の場合 (1)print n 2.1.2 そうでない場合 (1)illegal 判定条件はひとつ、実行文書はもうひとつ、ネストを下げる。

(8．4) PADによる表記例

■スライド(7．6)の問題3の解答例では、右図のような処理を行っています。

- HTMLを打ち出す部分は省略しています。
- この図は、下記URLよりダウンロードした、「Pad Simulator」にて作成しています。
  - http://www.vector.co.jp/soft/dl/win95/prog/se235608.html

■このPADは、処理の粗い流れを示しています。詳細なPADを作成する場合もあります。



(8．5)問題4

■右図のような掲示板の、処理方法をPADで描いてみてください。

■動作

- ページにアクセスすると、各メンバの名前と意見の一覧が、賛成(pro)と反対(con)とに分けて表示します。
- 自分の立場を表明するために、賛成(pro)か反対(con)かを入力するラジオスイッチを設けます。