プログラミング言語 II
④ Perl 入門(２)

阿萬裕久
aman@cs.ehime-u.ac.jp

パターンマッチング

- 前回説明(一部を演習)したように Perl 言語は**文字列処理を得意**としている
- 今回は文字列処理の中核を成す**パターンマッチング**について学ぶ
  - **特定の文字列**を探す
  - **特定の文字列パターン**(“abc”の繰り返し等)を探す
  - 文字列パターンを**正規表現**で与える

【例題１】

テキストファイルを入力とし，その中で "printf" という文字列が登場する行だけを表示するような Perl スクリプト find_printf.pl を作りなさい．ただし，行番号もあわせて出力すること．

```
$ perl find_printf.pl foo.c
9:  printf("N = %d¥n", N);
16:  printf("1 + ... + %d = %d¥n", N, sum);
```


【例題１】(答え)

```
$n = 0;
while ( $line = <> ){
    $n++;
    if ( $line =~ /printf/ ){
        print $n, ":", $line;
    }
}
```

(基本アルゴリズム)
- 行番号付きで内容を出力
- ただし，printf が登場する場合のみ

（解説）行番号付きで出力は前回と同じ

```
$n = 0;
while ( $line = <> ){
    $n++;
    if ( $line =~ /printf/ ){
        print $n, ":", $line;
    }
}
```

変数 n で行番号を管理しながら，
「一行ずつ読み込んでは出力する」の繰り返し

（解説）特定の文字列が登場するか？

```
$n = 0;
while ( $line = <> ){
    $n++;
    if ( $line =~ /printf/ ){
        print $n, ":", $line;
    }
}
```

変数 line には，各行の内容が入っている．
この中に「printf」が含まれる場合だけに限定したい．

（解説）パターンマッチング「=~」

- 変数の値に対し，所定のパターンがマッチする（見つかる）か？ → =~ で判定

```
if ( 変数 =~ /パターン/ ) {
    …
```

- （例）パターンマッチに成功する場合

```
$x = "this is a sample string";
if ( $x =~ /str/ ) {
    …
```

== だと値の完全一致になるが，
=~ だとパターンの一致になる

（解説）printf が登場する行のみ

```
$n = 0;
while ( $line = <> ){
    $n++;
    if ( $line =~ /printf/ ){
        print $n, ":", $line;
    }
}
```

正確には printf という文字列が一部に登場すれば
何でもよいので，sprintf という文字列にもマッチする

【例題2】

テキストファイルを入力として，その中で次のパターンが登場する行のみを表示する Perl スクリプト find_pattern.pl を作りなさい：
「**数字が 1 文字以上**登場し，その後ろに **ax または by** が登場する．途中に空白は入らない．」

```
$ perl find_pattern.pl foo.txt
8525ax
yyy089927byxxxx
```

```
8525ax
089927xxxx
yyy089927byxxxx
xyzby
```


【例題2】(答え)

```
while ( $line = <> ){
    if ( $line =~ /¥d+(ax|by)/ ){
        print $line;
    }
}
```

(基本アルゴリズム)
- **内容をそのまま出力**
- **ただし，指定パターンが登場する場合のみ**

(解説)「数字」というパターン

- 問題文では，0～9のうち，どの数字なのかは指定されていない
- かといって if 文を 10 個書くのは非現実的

- この場合 [0-9] という書き方が許されている
  (文字コードの並びで '0' から '9' までと考えた方が自然かも)
- 特別に ¥d でもよいことになっている

その他については例題2の解説ページを参照のこと

(解説)パターンマッチの繰り返し

- あるパターン P の繰り返しについて，正規表現では P^+ や P* といった書き方がある
  - P^+ は P を1回以上繰り返す
  - P* は P を0回以上繰り返す
- Perl で書く時も同様であり，パターンPに対して
  - **P+** は P を**1回以上**繰り返す
  - **P＊** は P を**0回以上**繰り返す

  また，その他に
  - **P?** は P を **1 回または0回**(つまり，Pを**省略可**)

（解説）「数字」が１文字以上

- これを正規表現で記述すると
  [0-9]+
  という書き方になるが，Perl では特別に ¥d でもよいので
  ¥d+
  と書いてもよい

（解答例から抜粋）

```
if ( $line =~ /¥d+(ax|by)/ ){
```


（解説）「ax または by」というパターン

- パターン P とパターン Q のいずれか一つという場合，(P|Q) という書き方をする

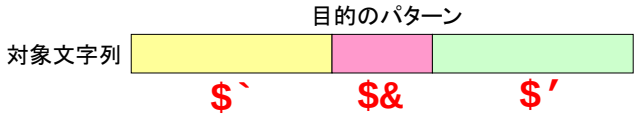
※途中に空白を入れないこと！（空白を入れると，それもパターンの一部と見なされる）

- よって，この場合は (ax|by) という書き方に

その他については
例題2の解説ページ
を参照のこと

（参考：課題2-1関連）パターンマッチした部分とその前後

- =~ によるパターンマッチングを行うと，その結果が自動的に三つの変数へ代入される
  - $& … パターンにマッチした部分
  - $` … パターンにマッチした部分より左
    （`はバッククォート：[Shift]+[@] で表示）
  - $' … パターンにマッチした部分より右
    （'はシングルクォート：[Shift]+[7] で表示）



【例題３】

テキストファイルを入力として，その中で **m で始まって s で終わる単語**のみを出力する Perl スクリプト find_word.pl を作りなさい．

- 実行例

```
$ perl find_word ex3.txt
managers
measures
metrics
......
```

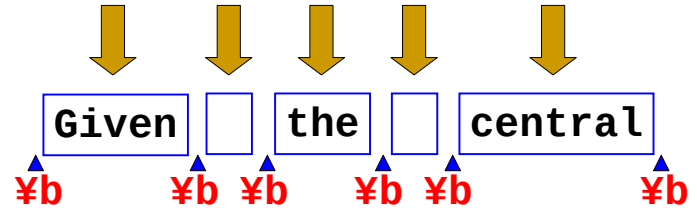
【例題３】(答え)

```
while ( $line = <> ){
    @words = split(/¥b/, $line);
    foreach $w ( @words ){
        if ( $w =~ /^m.*s$/ ){
            print $w, "¥n";
        }
    }
}
```


(解説)入力を単語に区切る

- Perl では特別な記号が用意されている: ¥b
- これは**単語の区切りにマッチ**する**長さ0**の特別な記号(※ここでいう単語＝アルファベット列)

```
split(/¥b/,"Given the central");
```



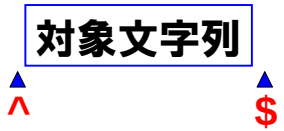
例題3の答え:対応部分

```
while ( $line = <> ){
    @words = split(/¥b/, $line);
    foreach $w ( @words ){
        if ( $w =~ /^m.*s$/ ){
            print $w, "¥n";
        }
    }
}
```

一行ずつ読み込み，その内容を単語分けして配列 words へ入れる

(解説)文字列の先頭と末尾

- 先ほどの ¥b と同様に，**文字列の先頭と末尾**をそれぞれ表した長さ0の特別な記号がある
  - ^ … 文字列の先頭
  - $ … 文字列の末尾



(例) printf という文字列にマッチさせる

```
/^printf$/
```

こうすれば sprintf にまでマッチすることはない

(解説)任意の文字

- m で始まり s で終わる文字列
  → `/^ms$/` でよいか？ ×:不十分！

- m と s の間には任意の文字が 0 個以上入ってもよい
- **任意の文字**を表す記号 `.`(ピリオド)
- よって，正解は `/^m.*s$/`

まとめ

- Perl **上での正規表現**によるパターンマッチングを学んだ
  - **繰り返し ＋ ＊**
  - **選択（｜）**
  - **行頭 ^**
  - **行末 $**
  - **任意の文字 .**
  - **単語の区切り ￥b**
- これらを使えば**高度な文字列処理も容易に実現**