プログラミング言語II
③ Perl 入門

阿萬裕久
aman@cs.ehime-u.ac.jp

スクリプト言語

- スクリプト言語では

プログラム
＝ コンピュータを動かすためのスクリプト（台本）

スクリプト

- 例）シェルスクリプト
  シェルに実行させたいコマンド列
- 例）Javascript
  ウィンドウ表示やフォーム処理の命令

簡易プログラミング言語という側面

- 厳密には，スクリプト言語とプログラミング言語の間に定義の違いは無い
- ただ，一般に「スクリプト言語」というと
  - **ある特定の用途に限っては簡潔に書ける**
  - **コンパイルは不要（インタプリタ方式）**

  というイメージがある
- 例えば，シェルスクリプトや Javascript でできることを **C 言語で書くのは大変**だが，**その逆に全てのことをシェルスクリプトでできるわけではない**

適材適所の考え方

- **汎用的な言語**（例えば，C, C++, Java）では，程度の差はあるが**様々な処理を実現**できる
  ただし，多くの労力や深い知識を必要とする場合も

- 一方，**スクリプト言語**は，**それぞれが得意とする処理**があり，その分野に関しては簡潔に書ける
  ※もちろん，汎用的な言語と遜色ない高度なスクリプト言語もある

適材適所に使い分けることが重要

スクリプト言語 Perl

- Perl:「パール」と読む
- 文字列(テキスト)処理を得意とする
- 構文は C 言語に似ている
- 実行方法:

  perl (スクリプト名)

  スクリプトの拡張子には .pl を使う
- 文字列処理が得意という特徴からCGIでも使われることが多い(最近は PHP の方が主流)

【例題1】

テキストファイルの内容を行番号付きで出力するフィルタプログラム cat_n.pl を作りなさい.

- 実行例

```
$ perl cat_n.pl foo.c
1: #include
2:
   (途中省略)
19: }
```

他のフィルタと同様に
perl cat_n.pl < foo.c
という使い方でも同じ結果に

【例題1】(答え)

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

この内容について, 順番に説明していきます

(基本アルゴリズム)

- 行番号を n で管理
- 一行読み込み, 行番号とその行の内容を出力

(解説)変数の扱い

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

$n という変数を用意し, 0 を代入する

Perl での変数:

1) 名前は $ で始まる
2) 宣言は不要
3) 型は無い(数字でも文字列でもよい)

型が無い？

- 変数の内容を「数値とするか，文字列とするか」はその時の演算で決まる

（例）

```
$x = “100”;
$x++;
$y = $x . “a”
```

変数 x に文字列「100」が入る

++ は数字を＋1するという演算なので，「100」は数字（百）と見なされ，x は「101」になる

. は文字列を連結するという演算なので，「101」は再び文字列と見なされ，y は「101a」になる

（解説）一行ずつ読み込む

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

$line という変数を用意し，そこに一行分の文字列をまるごと代入する while 文により，最後の行まで繰り返す

ダイヤモンド演算子 <>

- コマンドライン引数でファイルが指定されていれば，そのファイルが自動的に開かれ，＜＞はその内容に対応する

（例）

```
perl  cat_n.pl  foo.c
```

```
while ( $line = <> ){
   .....
}
```

自動的に foo.c がオープンされ，その内容が一行ずつ読み込まれる

ダイヤモンド演算子 <> （続き）

- コマンドライン引数が何も書かれていない時は，自動的に＜＞が標準入力に対応する

（例）

コマンドライン引数とは見なされない

```
perl  cat_n.pl  < foo.c
```

```
while ( $line = <> ){
   .....
}
```

自動的に標準入力から一行ずつ読み込まれる

コマンドライン引数と標準入力の両方を与えた場合

（例）

```
perl cat_n.pl foo.c < bar.c
```

- ダイヤモンド演算子＜＞では**コマンドライン引数が優先**： foo.c の内容が＜＞に対応
- **標準入力**（bar.c の内容）にアクセスしたいときは，<STDIN>と書く

（解説）出力

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

型を気にすることなく，表示したいものを列挙していく
複数ある場合はカンマで区切ってよい

【例題２】

テキストファイルを入力として，その内容を出力する Perl スクリプト hcat.pl を作りなさい．
ただし，最後の一行だけは出力しないものとする．

- 実行例

```
$ perl hcat.pl foo.c
#include
        （途中省略）
   return 0;
}
```

← 最後の一行だけは表示されない

【例題２】（答え）

```
@contents = <>;
pop(@contents);
foreach $line (@contents) {
    print $line;
}
```

この内容について，順番に説明していきます

（基本アルゴリズム）

- **全ての入力を配列 contents に入れて，末尾を削除**
- **配列の内容を一行ずつ出力**

（解説）配列

- 配列も変数と同様に**宣言不要**である
- 配列名は **@ で始まる**
  - @contents　…. 配列 contents（全体）
  - $contents[0]　…. 配列中の 1 番目の要素
  - $contents[1]　…. 配列中の 2 番目の要素
  - ……

※配列へ**まとめて代入**も可能

（例） @array = (“aa”, “bb”, “cc”);

これは $array[0] = “aa”; $array[0] = “bb”; $array[0] = “cc”;

【例題３】

CSV（カンマ区切り形式）のデータを読込み，その中の左から **3 列目と 2 列目だけを表示**する Perl スクリプト csv32.pl を作りなさい．
ただし，表示は "3列目−2列目" のかたちとする．

- 実行例

```
$ perl csv32.pl foo.csv
9-14
92-53
 （途中省略）
34-93
37-88
```

3列目 → 34-93　　2列目 → 92-53

【例題３】（答え）

```
@contents = <>;
foreach $line (@contents){
  @data = split(/,/, $line);
  print $data[2], "-", $data[1],
                （本当は前の行の末尾→）  "¥n";
}
```

この内容について，順番に説明していきます

（基本アルゴリズム）

- **全ての入力を配列 contents に入れ，**
- **一行ずつの内容をカンマ毎に切り出して出力**

（解説）split 演算子

- Unix コマンド cut を使うと，入力の中から特定の列だけを取り出すことができた

aaa,bbb,ccc ⇒ `cut -d , -f 2` ⇒ bbb

- Perl でも同様のこと（＋α）が実現できる：

split 演算子

`split (/,/, “aaa,bbb,ccc”);` ⇒

```
aaa
bbb
ccc
```

（解説）split 演算子　（続き）

- 与えられた**文字列**を所定の**パターン**で**分断**する
- よくある使い方

配列 = split (/パターン/, 文字列);

（例） @a = split (/-/, “state-of-the-art.”);

$a [0] = “state”;
$a [1] = “of”;
$a [2] = “the”;
$a [4] = “art.”;

（解説）カンマで分断

```
@contents = <>;
foreach $line (@contents){
  @data = split(/,/, $line);
  print $data[2], "-", $data[1],
                              "¥n";
}
```

$line の中には “24,14,9,26,83,53,18,55,75,69” といったかたちのデータが入っている
→ split 演算子によってカンマで分断され，
2 列目（14）は $data [1] に，3列目（9）は $data [2] に入る

まとめ

- **スクリプト言語**の概念（一般に言われているイメージ）を学んだ
  - **簡易プログラミング言語**
  - 得意とする処理分野を持つことが多い
  - インタプリタ型が多い
- **文字列処理を得意**とする Perl の基礎を学んだ
  - 構文は C に似ている
  - **変数は $xxx, 配列は @xxx**
  - フィルタ作成に便利なダイヤモンド演算子<>と split