ビギナーによるビギナー向けの真珠みがき　～基礎の基礎～

Kansai.pm 第2回セミナー
2000年5月20日
舟木 隆康　funaki@i.am

この時間の流れ

- いきなりソースを見て
- 結果を見てもらい
- 解説を行う
- 補足はあんまり詳しくしません　(^^;

- 判らない点は随時質問OK!
- それどころか、私が判らないところは他の人に解説して頂こうかな…　(^^;

この時間では

- スカラー変数とは
- スカラー変数に関する疑問
- SPRINTFとPRINTF
- 配列とは
- その他の配列操作法
- PUSHとPOP
- SHIFTとUNSHIFT
- 配列の中身すべてに何か行う
- FOREACH

スカラー変数とは

```
my $name = "Funaki";
my $age = 23;
my $weight = 64.8;


print "Mr. $name is $age years old and weighs $weight.¥n";
```

結果

```
Mr. Funaki is 23 years old and weighs 64.8.
```

スカラー変数とは

- $が先頭につく任意の名を持つ箱
- 文字、単語、文章、整数、小数点の値なら何でも入る箱（変数）である
  - シングルクォーテーション(')又はダブルクォーテーション（"）で囲まれていると文字列として代入する
  - 数値は「"」に囲まず、そのまま代入する
    - 囲んでも別に良いけどね…　(^^;

補足 (myについて)

- myってどんな意味？
  - myはスカラー変数や配列などを使う前に「今いるブロック内でのみ私は存在します」という宣言
  - サブルーチンを作ったときに、同じ名前のスカラー変数があるとバッティングしてしまい、想定しないエラーが出る可能性があるので宣言しとくと良い
  - サブルーチンが無い場合要らない　(^^;
    - つまりmy無しでいきなり「$hoge = 1;」としてもOK!

補足 (printについて)

- printって何？
  - printは出力する為の命令（関数）
  - 「”」か「'」で囲まれた文字列を出力
    - 「'」で囲まれていると「¥と'」以外の文字は全部まんま出力される
    - 「"」で囲まれている場合、いくつかの記号が特別な意味を持つ
  - 「"」に囲まれている中の「¥n」という記号は「改行しろ」の意味を持っている

スカラー変数に関する疑問

```
my $hoge = 5/3;
print "３人で割り勘すると値段は$hogeドルです¥n";

my $pai = 3.14159265358979323846264338327 9;
print "学校ではπを$paiで計算せよと教えられましたが、最近では$paiにしよう、
¥nという話が持ち上がっているようです。それでいーのか？ (^^;;¥n";
```

結果

```
３人で割り勘すると値段は1.66666666666667ドルです
学校ではπを3.14159265358979で計算せよと教えられましたが、最近では
3.1415926535899にしよう、
という話が持ち上がっているようです。それでいーのか？ (^^;;
```

- あらら…？ こういう時はどうすればいいの？

スカラー変数に関する疑問

```
my $hoge = 5/3;

$hoge = sprintf ("%4.2f", $hoge);
print "３人で割り勘すると値段は$hogeドルです¥n";

my $pai = 3.14159265358979323846264338327９;
printf ("学校ではπを%4.2fで計算せよと教えられましたが、最近では%dにしよう
、¥nという話が持ち上がっているようです。それでいーのか？　(^^;;¥n",$pai,$pai);
```

結果

```
３人で割り勘すると値段は1.67ドルです
学校ではπを3.14で計算せよと教えられましたが、最近では3にしよう、
という話が持ち上がっているようです。それでいーのか？　(^^;;
```

これで解決！

SPRINTFとPRINTF

- **sprintf**
  - スカラー変数の桁を切ったりする（フォーマット）したいときに使える関数
  - もちろんそれを受ける為のスカラー変数が必要
- **printf**
  - スカラー変数を出力するときにフォーマットしたいときに使う
  - 元のスカラー変数に影響は無い

配列とは

```
my @lang = ("java", "pascal", "fortran", "perl", "c", "cobol", "basic");

#めんどいなら
@lang = qw(java pascal fortran perl c cobol basic);

print $lang[3] . "¥n";
```

結果

```
perl
```

配列とは

- 配列はこのように代入する事もできる

```
my @hoge;

$hoge[0] = "java";
$hoge[1] = "pascal";
$hoge[2] = "fortran";
$hoge[3] = "perl";
...
$hoge[6] = "basic";
```

  - やっている事は同じ

配列とは

- @が先頭につく任意の名を持つ
- スカラー値をたくさん入れられる
- qwを使うと「"」と「,」が不要で楽

- 呼び出すときは $hoge[数字]

補足 (printについて2)

- print命令文の中でピリオド(.)を使うと、文字列をくっつける事ができる

```
print "Hello " . "world!";
print "Hello world!";
```

は同じ。

最初の"Hello "の後にスペースを入れるのを忘れないように

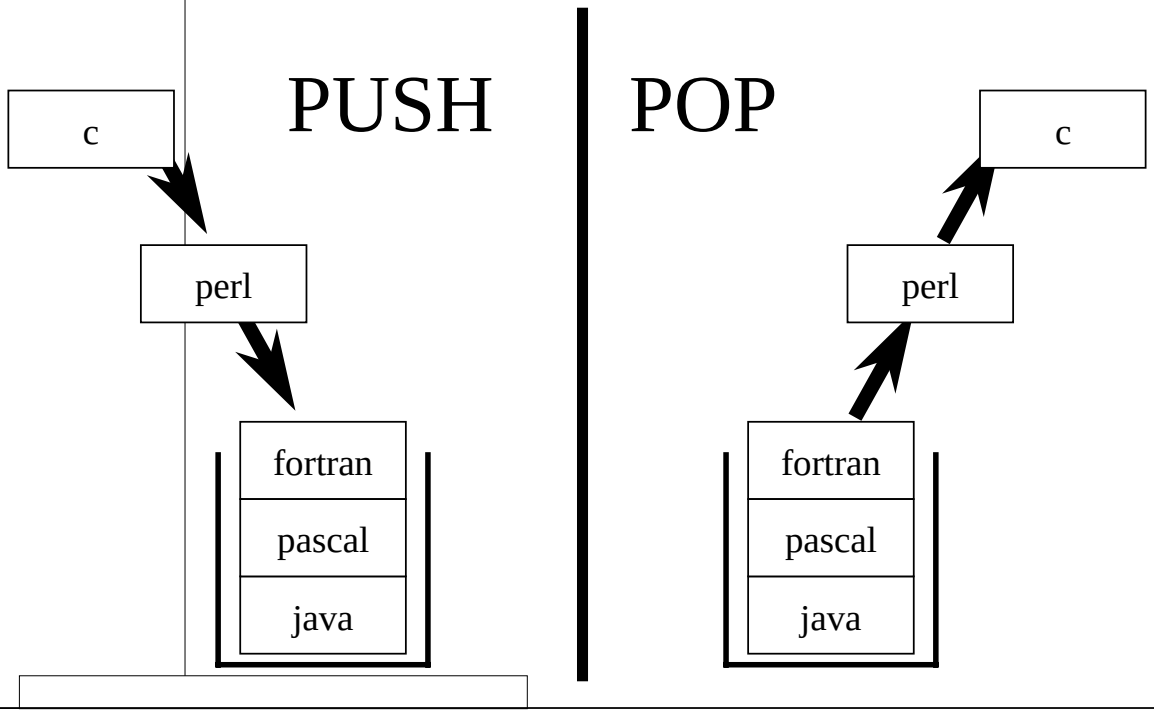
その他の配列操作法

```
my @lang;

push (@lang, "java");
push (@lang, "pascal");
push (@lang, "fortran");
push (@lang, "perl");
...
push (@lang, "basic");

print $lang[3] . "¥n";
```

結果

```
perl   #やっている事は前回の例と同じ
```

PUSHとPOP



その他の配列操作法

```
@lang = qw(java pascal fortran perl c cobol basic);

print pop (@lang) . "¥n";  #basicが飛び出してゆく
print pop (@lang) . "¥n";  #cobolが飛び出してゆく
print pop (@lang) . "¥n";  #cが...
print pop (@lang) . "¥n";
```

結果

```
basic
cobol
c
perl
```

その他の配列操作法

- 「push」と「pop」
  - 配列の最後に対して使われる
  - 配列をスタック(LIFO)として使える
    - pushすると最後尾に新たに追加
    - popすると最後尾から抜き出す
      - popすると抜き出された要素はなくなる

その他の配列操作法

```
@lang = qw(java pascal fortran perl c cobol basic);

print shift (@lang) . "¥n";
print shift (@lang) . "¥n";
print shift (@lang) . "¥n";
print shift (@lang) . "¥n";
```

結果

```
java
pascal
fortran
perl
```

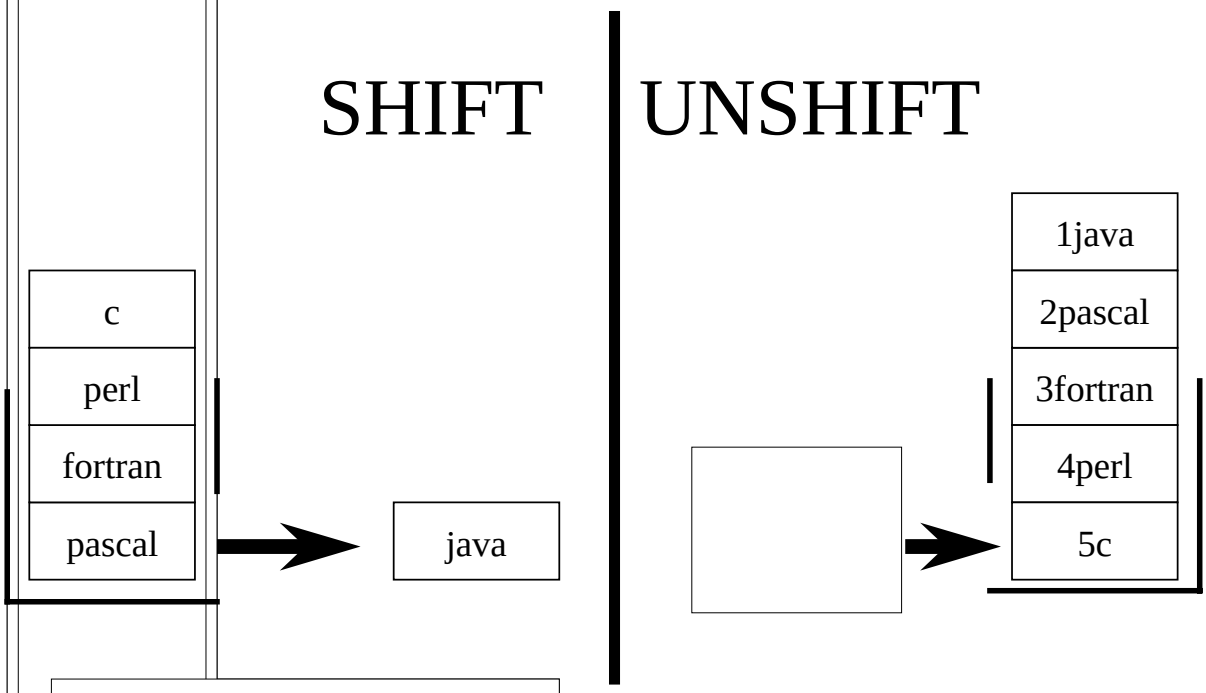
その他の配列操作法

```
my @lang;

unshift (@lang, "1java");          #判りやすさの為に
unshift (@lang, "2pascal");        #先頭に番号を振っ
unshift (@lang, "3fortran");       #てみました。
unshift (@lang, "4perl");
unshift (@lang, "5c");
unshift (@lang, "6cobol");
unshift (@lang, "7basic");

print "@lang";
```

結果

```
7basic 6cobol 5c 4perl 3fortran 2pascal 1java
```

SHIFTとUNSHIFT



その他の配列操作法

- 「shift」と「unshift」
  - 配列の先頭に対して使われる
  - shiftすると先頭から抜き出す
    - shiftすると抜き出された変数はなくなる
  - unshiftすると先頭に新たに追加
    - 追加する毎に古いのは後ろに行く
  - 配列をFIFOとして使える

配列の中身すべてに何か行う

```
@lang = qw(java pascal fortran perl c cobol basic);

foreach $i (@lang){
        print $i . "¥n";
}
```

結果

```
java
pascal
fortran
perl
c
cobol
basic
```

FOREACH

@lang = | java | pascal | fortran | perl | ・・・・ | cobol |

この配列に入っている項目一つ一つに何らかの処理を行いたい

foreach $i (@lang)すると…

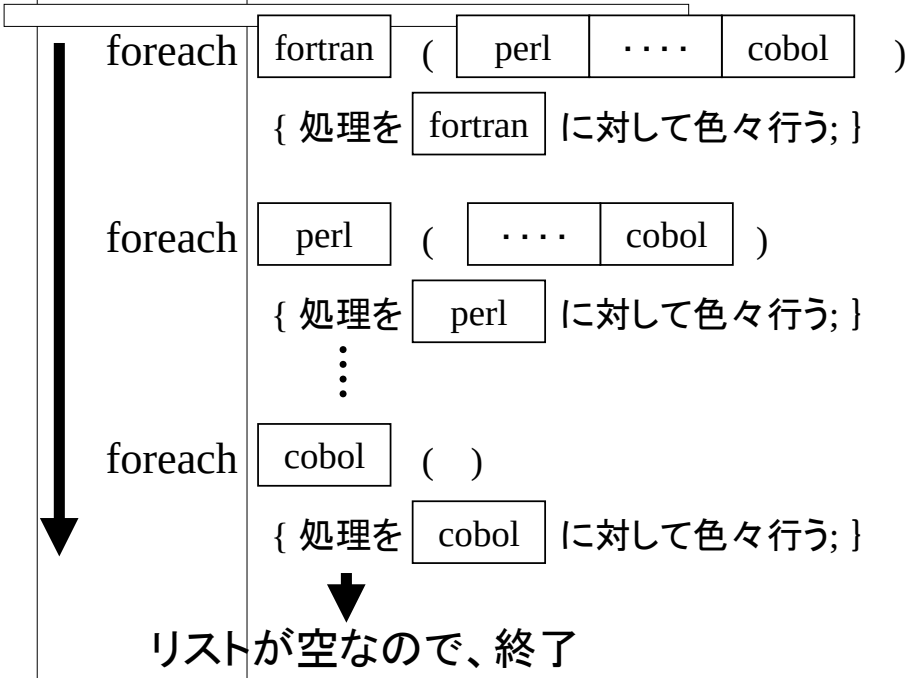
foreach | java | (| pascal | fortran | perl | ・・・・ | cobol |)

{ 処理を | java | に対して色々行う; }

foreach | pascal | (| fortran | perl | ・・・・ | cobol |)

{ 処理を | pascal | に対して色々行う; }

FOREACH



配列の中身すべてに何か行う

- 配列やリストの中身すべてを処理するのには**foreach文が便利**
  - 配列に対するめんどうな処理を$iという変数に一旦入れ、その$iに処理を行う
  - ブロック(「{」と「}」に挟まれている部分)の処理が終われば、最初に戻って、配列に入っている次の値を処理する
  - 元の配列の要素は$iに変更される

FOREACH

```
@lang = qw(java pascal fortran perl c cobol basic);

foreach $i (@lang){
  $i = uc($i);              # uc は小文字を大文字にする命令
  print $i . "¥n";
}
```

結果

```
JAVA
PASCAL
FORTRAN
PERL
C
COBOL
BASIC
```

Perlのいいトコ

- デフォルトが決まっている
  - スカラー変数$_;
  - 配列@_;

```
print;
print $_;           #上と同じ

foreach (@lang) { }
foreach $_ (@lang) { }              #上と同じ

uc;
uc $_;              ＃やっぱり上と同じ
```

Perlのいいトコ

```
@lang = qw(java pascal fortran perl c cobol basic);

foreach (@lang){            # foreach $_ (@lang) と同じ
  print uc . "¥n";           # print uc($_) . "¥n"; と同じ
}
```

結果

```
JAVA
PASCAL
FORTRAN
PERL
C
COBOL
BASIC
```

参考

- 文献
  - プログラミングPerl改訂版、オライリージャパン
  - 初めてのPerl第2版、オライリージャパン
  - Perl5パワフルテクニック大全集、インプレス

- ご意見やご質問があればどうぞ

ご静聴ありがとうございました

Kansai.pm第２回セミナー
ビギナーによるビギナー向けの
真珠みがき　～基礎の基礎～