map関数とgrep関数の覚書

2016/03/27
ループをゴリゴリ書かないで済ますため雛型用に記録します。

map関数

map関数使用例

map.pl

```
use strict;

my @data = ( 1, 2, 3, 4, 5 );
my %hash = ( 1=>"a", 2=>"b", 3=>"c", 4=>"d", 5=>"e" );
my @ans1;
my @ans2;

sub func
{
  my ($arg) = @_;

  return $arg / 2;
}

  @ans1 = map( $_ * 2   , @data );          ## exsample 1
  @ans2 = map( func($_) , @data );          ## exsample 2
  map( { $hash{$_}=uc($hash{$_}) } @data ); ## exsample 3

  print "ex 1 : " . join(", ", @ans1) . "\n";
  print "ex 2 : " . join(", ", @ans2) . "\n";
  print "ex 3 : " . join(", ", values(%hash)) . "\n";
```

実行結果

```
$ perl map.pl
ex 1 : 2, 4, 6, 8, 10
ex 2 : 0.5, 1, 1.5, 2, 2.5
ex 3 : B, E, C, A, D
$
```

map関数に与える配列の各要素に計算(ex1, ex2)を行い結果をリストで返します。
結果がいらないなら各要素毎に何か処理(ex 3)を行わせるためだけに使う事もできます。

grep関数

grep関数使用例

grep.pl

```
use strict;

my @data1 = ( 1, 2, 3, 4, 5, 2, 4, 6 );
my @data2 = ( "apple", "orange", "potato", "grape" );
my @ans1;
my @ans2;
my @ans3;
my @ans4;

sub func
{
  my ($arg) = @_;

  return $arg % 2;
}

  ## main

  @ans1 = grep( $_ % 3   ,  @data1 );                              ## exsample 1
  @ans2 = grep( func($_) ,  @data1 );                              ## exsample 2
  @ans3 = grep( { 1 if (($_==3)||($_==5)||($_==6)) } @data1 ); ## exsample 3
  @ans4 = grep( /[gr]/   ,  @data2 );                              ## exsample 4

  print "ex 1 : " . join(", ", @ans1) . "\n";
  print "ex 2 : " . join(", ", @ans2) . "\n";
  print "ex 3 : " . join(", ", @ans3) . "\n";
  print "ex 4 : " . join(", ", @ans4) . "\n";
```

実行結果

```
$ perl grep.pl
ex 1 : 1, 2, 4, 5, 2, 4
ex 2 : 1, 3, 5
ex 3 : 3, 5, 6
ex 4 : orange, grape
$
```

grep関数に与える配列の各要素に判定処理(ex1, ex2)を行い結果が真なら要素をリストで返します。
判定処理は何か処理(ex 3)でも構わないし、配列が文字列であれば正規表現(ex 4)を適用する事もできます。

map,grep関数応用

配列Aの重複要素を排除

samle1.pl

```
$ cat sample1.pl
use strict;

my @aryA = ( "radish",     "banana", "orange",
```

```
              "apple",      "ginger", "banana",
              "strawberry", "grape",  "apple" );
my $str0;
my $str1;

  ## 配列A
  $str0 = join(", ", @aryA);

  ## 配列Aから重複排除
  {
    my %ans;
    map( { $ans{$_}=0 } @aryA );

    $str1 = join(", ", keys(%ans) );
  }

  print "0:$str0\n";
  print "1:$str1\n";
```

実行結果

```
$ perl sample1.pl
0:radish, banana, orange, apple, ginger, banana, strawberry, grape, apple
1:grape, orange, strawberry, banana, radish, apple, ginger
$
```

配列Aの要素をハッシュのキーにして重複を消します。map関数をループ処理の代替としています。

処理	map関数の時

my %ans;
foreach (@aryA)
{
  $ans{$_}=0;
}
@ans = keys(%ans);
	my %ans;
	map({ $ans{$_}=0 } @aryA);
	@ans = keys(%ans);

このくらいだとforeachを1行で書いてしまえばいいので微妙かな。

配列Bの要素を配列Aから削除

sample2.pl

```
use strict;

my @aryA = ( "radish",      "banana", "orange",
             "apple",      "ginger", "banana",
             "strawberry", "grape",  "apple" );
my @aryB = ( "radish",      "ginger", "potato" );
my $str0;
```

```
my $str1;

  ## 配列A
  $str0 = join(", ", @aryA);

  ## 配列Aから配列Bの要素排除
  {
    my %ex  =  map(  ($_=>0) ,        @aryB );
    my @ans = grep( !exists($ex{$_}), @aryA );

    $str1 = join(", ", @ans );
  }

  print "0:$str0\n";
  print "1:$str1\n";
```

実行結果

```
$ perl sample2.pl
0:radish, banana, orange, apple, ginger, banana, strawberry, grape, apple
1:banana, orange, apple, banana, strawberry, grape, apple
$
```

配列Bの要素をハッシュのキーにして、配列Aの要素がハッシュのキーとして存在するか否かで排除有無を判定しています。

処理	map,grep関数組み合わせ

my %ex;
my @ans;
foreach (@aryB)
{
  $ex{$_}=0;
}
foreach (@aryA)
{
  if (!exists($ex{$_}))
  {
    push(@ans, $_);
  }
}
	my %ex = map(($_=>0) , @aryB);
	my @ans = grep(!exists($ex{$_}), @aryA);

まだこのくらいならどちらでも意図は伝わりそう。

重複要素と配列Bの要素を配列Aから削除

map関数	grep関数

sample3.pl
use strict;
 	
my @aryA = ("radish", "banana", "orange",
"apple", "ginger", "banana",
"strawberry", "grape", "apple");
my @aryB = ("radish", "ginger", "potato");
my $str0;
my $str1;
 	
##配列A
$str0 = join(" , ", @aryA);
 	
##配列Aから重複排除後に配列Bの要素を排除
{
my %ans = map(($_=>0), @aryA);
map({ delete($ans{$_}) } @aryB);
 	
$str1 = join(" , ", keys(%ans));
}
 	
print "0:$str0\n";
print "1:$str1\n";
	sample4.pl
	use strict;
	
	my @aryA = ("radish", "banana", "orange",
	"apple", "ginger", "banana",
	"strawberry", "grape", "apple");
	my @aryB = ("radish", "ginger", "potato");
	my $str0;
	my $str1;
	
	##配列A
	$str0 = join(" , ", @aryA);
	
	##配列Aから重複排除後に配列Bの要素を排除
	{
	my %wk;
	my @ans;
	my %ex = map(($_=>0), @aryB);
	
	@ans = grep(!$wk{$_}++, @aryA);
	@ans = grep(!exists($ex{$_}), @ans);
	
	$str1 = join(" , ", @ans);
	}
	
	print "0:$str0\n";
	print "1:$str1\n";

実行結果

sample3.pl	sample4.pl
$ perl sample3.pl 0:radish , banana , orange , apple , ginger , banana , strawberry , grape , apple 1:banana , strawberry , apple , grape , orange $	$ perl sample4.pl 0:radish , banana , orange , apple , ginger , banana , strawberry , grape , apple 1:banana , orange , apple , strawberry , grape $

関数なのに関数として使っていない例ばっかりで……

処理	map関数処理	grep関数処理

my %ans;
foreach (@aryA)
{
  $ans{$_}=0;
}
foreach (@aryB)
{
  delete($ans{$_});
}
@ans = keys(%ans);
	my @ans;
	my %ans = map(($_=>0), @aryA);
	map({ delete($ans{$_}) } @aryB);
	@ans = keys(%ans);
		my %wk;
		my @ans;
		my %ex = map(($_=>0), @aryB);
		@ans = grep(!$wk{$_}++, @aryA);
		@ans = grep(!exists($ex{$_}), @ans);

map関数＋ハッシュの組み合わせでもgrep関数＋ハッシュの組み合わせでも実現できます。処理の意味合いによってどちらを使うか考えてみるのが良いかなと思います grep関数を使う場合は、要素の並びが変わらないので、できるだけ並びを維持したいときはgrep関数が良いかもしれません。

技術資料, Perl, map, grep, 関数