Outils informatiques

2. Les références en Perl

DESS TEXTE

Introduction (1)

- Perl dispose des 3 types élémentaires suivants :
  - les scalaires (notés $scalaire),
  - les listes (notées @liste),
  - les tableaux associatifs (notés %hash).
- De plus, sauf indication contraire, Perl manipule les variables par leur contenu.
- Une conséquence de cela : Perl ne manipule que des listes plates.

Problème : que se passe-t-il lorsque l'on souhaite appeler une fonction en passant deux listes en arguments ?

Introduction (2)

Exemple :

```
@radicaux = ("aimer", "manger", "donner");
@terminaisons = ("ai", "as", "a", "ons", "ez", "ont");
@futur = &conjuguer(@radicaux,@terminaisons);
print  @futur ;

sub  conjugue {
  my (@r, @t) = @_ ;
  my ($v,$s,$i) ;
  my @pro= ("je", "tu", "elle", "nous", "vous", "elles")
  my @res ;
  foreach $v (@r) {
    $i= 0 ;
    foreach $s (@t) {
      push(@res, $pro[$i]." ".$v.$s) ; $i++ ;
    }
  }
  return @res ;
}
```

Introduction (3)

- ici *print @futur* renvoie la liste vide, car le symbole @_ définit une liste “plate” (radicaux et terminaisons concaténés).

- Comment faire pour que **@r** et **@t** contiennent bien les éléments des listes *@radicaux* et *@terminaisons* respectivement ?

- Solution : passer les arguments de la fonction *conjugue* par **référence**.

Notion de Référence

- Lors du passage de paramètres par référence, on ne manipule plus une variable mais l'adresse de cette variable (adresse qui est un scalaire !).

- Idée : pouvoir désigner une liste ou un tableau associatif de manière non-équivoque au moyen d'un scalaire.

- Questions :
  1. comment créer une référence à un tableau ?
  2. comment traiter cette référence (notion de déréférencement) ?

Création de références

- Il existe deux façons de créer une référence en Perl :
  - **Règle 1 :** placer un \ devant la variable pour laquelle on désire créer une référence, exemples :

```
$ref_liste = \@liste ;
$ref_hash = \%hash ;
```

  - **Règle 2 :** utilisation des références anonymes, exemples :

```
$ref_liste = ["toto", 2, "mot"] ;
$ref_hash = { "titre" => "Tintin",
              "auteur" => "Hergé" } ;
```

Déréférencement (1)

- A présent, nous savons fabriquer l'*adresse* d'une liste ou d'un tableau.

- Il existe deux façons d'utiliser cette adresse en Perl :

  - **Règle 1 :** écrire {$la_reference} partout où on devrait mettre le nom du tableau, exemples :

```
@{$ref_liste}    au lieu de   @liste
${$ref_liste}[0] au lieu de   $liste[0]
```

  - Autres exemples :

Action	Via tableau	Via référence
désigner le tableau a	@a	@{$ref_a}
affecter le 1er élément	$a[0]=1	${$ref_a}[0]=1
trier le tableau	sort(@a)	sort(@{$ref_a})

Déréférencement (2)

- (suite)
  - Pour les tables de hachage :

Action	Via hash	Via référence
désigner la table de hachage a	%a	%{$ref_a}
affecter un élément	$a{'jour'}=1	${$ref_a}{'jour'}=1
récupérer les clés	keys(%a)	keys(%{$ref_a})

  - **Règle 2 :** écriture simplifiée, écrire **->** au lieu de **${...}**, exemples :

```
$ref_liste->[0]      remplace ${$ref_liste}[0]
$ref_hash->{'toto'} remplace ${$ref_hash}{'toto'}
```

- **Attention** : ne pas confondre $ref_hash{'toto'} et $ref_hash->{'toto'} !

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions

- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions
- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;
- Il s'agit d'un tableau contenant 3 références à des tableaux

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions
- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;
- Il s'agit d'un tableau contenant 3 références à des tableaux
- ici : `${$a[1]}[2]` =

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions
- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;
- Il s'agit d'un tableau contenant 3 références à des tableaux
- ici : `${$a[1]}[2] =  6`

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions

- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;

- Il s'agit d'un tableau contenant 3 références à des tableaux

- ici : `${$a[1]}[2]` = 6
ce qui se note aussi : $a[1]->[2], où encore $a[1][2]

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions

- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;

- Il s'agit d'un tableau contenant 3 références à des tableaux

- ici : `${$a[1]}[2]` = 6
ce qui se note aussi : $a[1]->[2], où encore $a[1][2]

- généralisation : $a[LIGNE][COLONNE]

Déréférencement (3)

- Cas particulier d'utilisation de la référence : les tableaux à 2 dimensions

- Exemple : @a = ([1, 2, 3] , [4, 5, 6], [7, 8, 9]) ;

- Il s'agit d'un tableau contenant 3 références à des tableaux

- ici : `${$a[1]}[2]` = 6
ce qui se note aussi : $a[1]->[2], où encore $a[1][2]

- généralisation : $a[LIGNE][COLONNE]

- valable également pour les tableaux à plus de 2 dimensions

Utilisation des références

- Exercice :

  Réécrire le programme de conjuguaison
  des verbes "aimer", "manger", "donner"
  au futur en utilisant une fonction.

Remarques (1)

- Il est possible de créer des références sur tout type de données (scalaires, fonctions, etc).

- Lorsque le nom de la référence est atomique, lors du déréférencement, on peut ommettre les accolades :

```
@{$reference}     équivaut à  @$reference
${$reference}[0]  équivaut à  $$reference[0]
```

- Pour vérifier si une variable contient une référence, on utilise la fonction *ref($var)* .

- Cette fonction *ref* renvoie le type de l'objet référé : HASH ou ARRAY par exemple.

Remarques (2)

- Si vous essayez d'utiliser une variable contenant une référence sans l'avoir d'abord déréférencé, vous obtenez un message du style :
  ARRAY(0x80f5dec) ou HASH(0x826afc0)

- Vous pouvez vérifier si deux références réfèrent au même type via == ou *eq*.

- Pour plus de renseignements sur les références, voir la documentation Perl nommée **perlref**.

Conclusion

- En Perl, l'utilisation des références permet de définir des structures complexes, représentant des problèmes divers.
- L'utilisation des références respecte 4 règles : deux de référencement et deux de déréférencement.
- La notion de *référence* est proche de celle de *pointeur* en langage C.