Perl TD2 : Tableaux associatifs (ou hachages)

Résumé sur les tableaux associatifs

Les tableaux associatifs (ou hachages), comme les tableaux classiques, sont une collection de données, mais les indices repérant les éléments ne sont pas forcément des entiers, mais des valeurs quelconques que l"on nomme clés. L"ordre des couples (clé/élément) dans un hachage est imprévisible. Il ne dépend que de la manière dont Perl les stocke pour faciliter sa recherche en fonction des clés. De plus, l'accès à un hash inexistant implique sa création. Le nom d'une variable de type hachage commence par % (Rq : les variables %hash, @hash et $hash n'ont rien à voir entre elles).

```
# Assigner un element (crée le hachage s'il n"existe pas)
my %hash;
$hach{"aaa"} = 3; # cle = "aaa", valeur = 3
$hach{5.6} = "abc"; # cle = "5.6", valeur = "abc"
# Initialisation globale ou copie de hachage
%hach = ("aaa",3,5.6,"abc"); # idem que ci-dessus
#ou encore
%hach = (aaa => 3, 5.6 => abc); # idem que ci-dessus
%cop = %hach; # recopie de hachage
```

Fonction **keys** : fournit la liste de toutes les clés d'un hachage. Utilisée avec une boucle foreach, elle permet de parcourir tous les éléments du hachage (sans bien sûr maîtriser l'ordre du parcours) :

```
@liste = keys(%hach); # @liste = ("aaa", 5.6) ou (5.6, "aaa")
for $cle (keys(%hach)) {
    print "cle: $cle, valeur: $hach{$cle}\n";
}
```

Fonction **values** : fournit la liste de tous les éléments d"un hachage, dans le même ordre que la liste des clés renvoyées par keys :

```
@liste = values(%hach); # @liste = (3,"abc") ou ("abc",3)
```

Fonction **each** : fournit un couple (clé/valeur) sous forme d"une liste à 2 eléments. A chaque application sur le même hachage, le couple suivant est renvoyé (liste vide si c'est fini) :

```
while ( ($cle,$val)=each(%hach) ) {
    print "cle: $cle, valeur: $val\n";
}
```

Fonction **delete** : permet de supprimer un élément d"un hachage :

```
delete $hach{"aaa"}; # %hach ne contient plus le couple cle/element: "aaa"/3
```

Expressions régulières.

les expressions régulières permettent de manipuler un texte de façon très puissante. On les utilise en particulier pour tester si une chaîne de caractères comporte un certain motif.

Pour trouver une sous-chaîne :
$m/\$exprreg/$: opérateur de recherche (match). Le 'm' est facultatif.
La recherche s'applique à `$_` par défaut sinon il faut utiliser l'opérateur `=~`

$/\$exprreg/i$: avec l'option i, recherche non sensible à la casse.
Le '.' remplace n'importe quel caractère, le '^' désigne le début de chaîne, le '$' la fin de chaîne.
Dans une expression régulière, on peut utiliser les raccourcis suivants :

- `\d` : un chiffre (équivalent à `[0-9]`)
- `\D` : tout sauf un chiffre (équivalent à `[^0-9])`
- `\w` : un caractère alphanumérique (équivalent à `[0-9a-zA-Z_]`)
- `\W` : tout sauf un caractère alphanumérique (équivalent à `[^0-9a-zA-Z\_])`
- `\s` : un espacement, équivalent à `[ \n\t\r\f]`
- `\S` : tout sauf un espacement, équivalent à `[\^ \n\t\r\f]`

Perl affecte 3 variables spéciales : $&, $‘ et $’, qui contiennent respectivement la partie de la chaine qui a “matché”, celle qui est avant et celle qui est après. Ainsi la position “matchante” est $length(\$`) + 1$.

Pour récupérer certaines parties de la chaîne qui matchent des sous-expressions du motif, on place ces sous-expressions entre parenthèses. Les différents résultats sont stockés dans des variables `$1`, `$2`, ...
Ex :

```
if ($time =  /(\d\d):(\d\d):(\d\d)/) { # heure au format hh:mm:ss
   $hours = $1;
   $minutes = $2;
   $seconds = $3;
}
```

1 Exercice 1

La table suivante spécifie la maladie associée à chaque gène :

AD4	Alzheimer's Disease
BRCA1	Breast Cancer
DMD	Duchenne Muscular Dystrophy
FMR1	Fragile X Syndrome
GBA	Gaucher's Disease

Ecrivez un script Perl qui crée une table de hachage dont les clés sont les éléments de la première colonne et les valeurs les éléments de la deuxième colonne. Le script demandera ensuite à l'utilisateur d'entrer un nom de gène, et il affichera la maladie correspondante. Le programme devra afficher un message d'erreur si le gène entré n'est pas trouvé dans la table.
Remarque : pour définir la table de hachage, vous pouvez utiliser ces 2 syntaxes :

```
%genes = ("AD4", "Alzheimer's disease", "BRCA1",
          "Breast Cancer",...);
```

OU

```
%genes = (AD4 => "Alzheimer's disease",
          BRCA1 => Breast Cancer,
          ...);
```

2 Exercice 2

Le fichier **kinases_map.txt** a été récupéré depuis la banque OMIM, il contient un tableau d'informations sur les gènes de protéines kinases de l'homme. Chaque ligne contient les informations sur un gène, les différents champs sont séparés par une barre verticale. La description des champs est :

— Gene symbol(s),
— Gene name,
— date of entry to the database,
— Cytogenetic location (chromosome number followed by cytogenetic band(s)),
— Accession number in OMIM.

Ecrivez un script Perl qui construit une table de hachage dont les couples clés/valeurs sont les données 'Accession number in OMIM'/'Gene name' trouvés dans le fichier.

3 Exercice 3

Ecrivez un script perl qui demande à l'utilisateur de saisir une chaine de caractères et teste si elle contient :

1. le mot **kinase** au singulier ou au pluriel,
2. le motif **ATAT** répété au moins une fois,
3. le motif **CGCG** répété 2 fois, suivi par trois symbols différents de **A** et de **C**,
4. le motif **ACG** suivi de 3 caractères parmi **A**, **C**, **G** et **T**, suivi de **T** répété entre 4 et 6 fois.

4 Exercice 4

On dispose d'un fichier contenant sur chaque ligne les renseignements suivants, séparés par le caractère **:**

— numéro d'étudiant composé de huit chiffres,
— un nom en majuscules,
— un prénom (simple ou composé) en minuscules, sauf la ou les initiale(s) et pouvant comporter des accents,
— un code composé de majuscules et de chiffres.

Ecrivez un script perl qui construit une table de hachage dont les clés sont les numéros d'étudiants et les valeurs les adresses email des étudiants. L'adresse email est constituée du prénom tout en minuscules et sans accent (avec des **-** si c'est un prénom composé), suivi d'un point, suivi du nom en minuscules, suivi de la chaine de caractères **@u-bordeaux.fr**.
Le script affichera la liste des adresses email des étudiants.

5 Exercice 5

Considérez un fichier contenant un texte quelconque.

1. Ecrivez un script Perl qui compte le nombre total d'occurences de chaque mot du texte. Si le texte est "les coccinelles dévorent les pucerons", vous devez obtenir le résultat :

   ```
   2  les
   1  coccinelles
   1  dévorent
   1  pucerons
   ```

   indice : expérimentez l'opérateur `each`.

2. Affichez également le nombre total de mots du texte, ainsi que le nombre de mots différents (i.e. chaque mot doit être compté une seule fois).
   Pour l'exemple précédent :

   ```
   Nb total mots: 5
   Nb mots differents: 4
   ```

   indice : expérimentez les opérateurs `keys` et `values`.

3. Trouvez une autre solution qui n'utilise pas l'opérateur values.