Chapitre 5

Le langage de programmation Perl

On peut se demander ce que vient faire ce chapitre sur un langage de programmation en plein milieu d'un livre consacré aux sites Web. En fait, Perl est le langage privilégié associé à CGI, que nous verrons dans le chapitre suivant et il sert, de nos jours, principalement à cela. Donnons donc quelques notions sur ce langage.

Perl est un langage de programmation créé par Larry Wall en 1987 qui reprend des fonctionnalités du langage C, du langage BASIC et des langages de commandes *sed*, *awk* et *shell* (*sh*). C'est un langage interprété particulièrement adapté au traitement et à la manipulation de fichiers texte, notamment du fait de l'intégration des expressions régulires dans la syntaxe du langage.

Les deux livres de référence sont [SC-98] pour une introduction et [WCS-96] pour une étude complète.

5.1 Installation et premier programme

5.1.1 Installation

Le site officiel de Perl est :

`http://www.perl.org/`

Sur la page d'accueil, il y a un bouton `Download Perl`, sur lequel on cliquera, ce qui renvoie à une page qui propose des binaires pour plusieurs systèmes d'exploitation, en particulier pour *Windows* qui nous intéresse ici. Choisissons, par exemple, `Download StrawberryPerl`, qui renvoie à la page d'accueil du projet correspondant :

`http://strawberryperl.com/`

En cliquant sur `Download Strawberry Perl 5.12.3.0` (dernière version stable à la date où nous écrivons ce chapitre), on télécharge le fichier :

`strawberry-perl-5.12.3.0.msi`

que l'on peut placer dans le répertoire que nous voulons.

En double-cliquant sur ce fichier téléchargé, on installe l'interpéteur perl, par exemple dans le répertoire :

```
E:\programme\strawberry
```

On pourra remarquer en particulier un sous-répertoire `perl`, contenant lui-même un sous-répertoire `bin`, contenant un fichier `perl.exe`.

5.1.2 Premier programme

Écrivons notre premier programme Perl, que nous sauvegardons sous le nom `test.pl` (le suffixe `.pl` est traditionnel pour les programmes Perl mais pas obligatoire) :

```
#!E:/programmes/strawberry/perl/bin/perl.exe

print "Bonjour";
```

Il suffit, dans une invite de commande d'écrire '`test.pl`' puis d'appuyer sur la touche retour pour voir apparaître 'Bonjour'.

Commentaires.- 1^o) La première ligne du programme Perl est obligatoire. Elle doit indiquer clairement l'emplacement de l'interpréteur Perl.

- 2^o) Comme en langage C, les instructions Perl se terminent par un point-virgule.

- 3^o) L'affichage s'effectue grâce à la fonction (prédéfinie) `print`. On remarquera que, par rapport au langage C, il n'y a pas de couple de parenthèses.

- 4^o) Les constantes chaînes de caractères peuvent s'écrire comme en langage C, encadrées par des guillemets verticaux (comme dans notre exemple) mais ces guillemets verticaux peuvent être remplacés ici par des apostrophes verticales si on préfère. Nous verrons la différence de signification entre guillemets et apostrophes ci-dessous.

5.2 Premiers éléments de langage Perl

Nous supposons que le lecteur sait programmer en langage C. Nous revoyons rapidement les éléments d'un langage de programmation en indiquant quelle en est la syntaxe en Perl.

5.2.1 Commentaires

Syntaxe.- Toute ligne commençant par '`#`' est considérée comme une ligne de commentaire, non analysée par l'interpréteur Perl.

La première ligne, commençant par `#!` est un peu spéciale et ne fait pas partie de Perl proprement dit. C'est la convention UNIX pour spécifier l'emplacement d'un interpréteur de lignes de commande.

Exemple.- Il est traditionnel en particulier de commencer un programme Perl par une courte description de ce qu'il fait. Notre premier exemple devrait en fait être écrit :

```
#!E:/programmes/strawberry/perl/bin/perl.exe
#
# test.pl - un test pour voir si Perl est present
#
print "Bonjour";
```

5.2.2 Scalaires

Les entités entières, réelles, caractères et chaînes de caractères sont appelées des **scalaires** en langage Perl. Comme en langage C, il n'existe pas de type booléen : on utilise le type entier avec 0 comme faux et 1 comme vrai.

Constantes.- Les constantes sont formées comme en langage C, que ce soit des constantes entières (123 ou -12 par exemple), réelles (1.2 ou 1.3e-5), caractères ('a') ainsi que chaînes de caractères ('Le petit garçon joue au ballon') avec, pour ces dernières, des apostrophes verticales pour les encadrer. Comme en langage C, Perl reconnaît un jeu de 256 caractères.

Opérations.- On utilise les signes '+' pour l'addition, '*' pour la multiplication, '-' pour la soustraction, '/' pour la division, '**' pour l'élévation à la puissance et '%' pour modulo.

L'application de l'opération '%' à des réels commence par transformer ceux-ci en entiers (leur valeur entière). Ainsi `25.3 % 4.43785` est d'abord converti en `25 % 4`, ce qui donne l'entier 1.

Types.- Il n'y a pas de typage explicite en langage Perl. Le type (implicite) est déterminé par la forme de la constante et, plus généralement, de l'expression qui lui est attribuée.

Variables.- Les identificateurs sont formés comme en langage C. Une **variable scalaire** est un identificateur précédé du caractère '$'. On peut donc écrire :

```
#!E:/programmes/strawberry/perl/bin/perl.exe
#
# Test des variables scalaires
#
$n = 100;
$fruit = 'pomme',
print '$n grammes de $fruit';
```

```
print "\n$n grammes de $fruit";
```

Ce programme nous permet de faire la différence entre les apostrophes et les guillemets : la première instruction d'affichage affiche exactement ce qui se trouve entre les apostrophes alors que dans la seconde, d'une part, la séquence d'échappement est interprétée et, d'autre part, les variables sont remplacées par leurs valeurs.

Expressions booléennes.- Pour obtenir un booléen, on peut :

- utiliser les entiers, à titre de constante booléenne ;
- comparer deux expressions scalaires numériques grâce aux opérateurs `==`, `!=`, $<$, $<=$, $>$ et $>=$;
- comparer deux expressions scalaires chaîne de caractères grâce aux opérateur `eq` pour l'égalité (*EQual* en anglais), `ne` pour la diséquation (*Not Equal* en anglais), `gt` pour strictement plus grand dans l'ordre lexicographique (*Greater Than* en anglais), `ge` pour plus grand ou égal (*Greater or Equal* en anglais), `lt` pour strictement plus petit (dans l'ordre lexicographique, (*Lesser Than* en anglais) et `le` pour plus petit ou égal (*Lesser or Equal* en anglais) ;
- combiner des expressions booléennes grâces aux connecteurs logiques de négation `not` (ou `!` comme en langage C), de conjonction `and` (ou `&&`) et de disjonction `or` (ou ||).

5.2.3 Entrées et sorties standard

Nous avons déjà vu la fonction `print()` pour l'affichage. Pour une entrée à partir du clavier, on utilise la variable spéciale <`STDIN`>, comme dans le programme ci-dessous :

```
#!E:/programmes/strawberry/perl/bin/perl.exe
#
# bonjour.pl
#
print "Nom : ";
$nom = <STDIN>;
print "Bonjour $nom\n";
```

5.2.4 Structures de contrôle

Instruction primitive.- Comme en langage C, une instruction primitive (affectation ou appel d'une procédure) se termine par un point-virgule ‘ ;’.

Séquencement.- Comme en langage C, un séquencement est une suite d'instructions, encadrée par des accolades ‘{’ et ‘}’ lorsqu'il y a plus de deux instructions.

Test, alternative et test multiples.- Comme en langage C, la syntaxe d'un test est :

```
if (boolean) instruction
```

et celle d'une alternative est :

```
if (boolean) instruction1 else instruction2
```

Mais on peut aussi utiliser :

```
if (boolean1)      instruction1
elseif (boolean2)  instruction2
elseif (boolean3)  instruction3
```

Itérations.-

5.2.5 Modularité

5.3 Structures de données

5.3.1 Tableaux et listes

En Perl, un tableau peut être considéré comme un ensemble ou comme une liste.

Nom.- Le nom d'un tableau est un identificateur précédé du caractère '@'.

Constante tableau.- Un tableau est une liste de valeurs séparées par des virgules et encadrées par des parenthèses :

```
@chiffres = (1,2,3,4,5,6,7,8,9,0);
@fruits = ('amande','fraise','cerise');
@alphabet = ('a'..'z');
@a = ('a');
@nul = ();
@cartes = ('01'..'10','Valet','Dame','Roi');
```

Les deux points signifient de « tant à tant ».

Référence à un élément.- Le i-ième élément (le premier indice étant, comme en langage C, 0) du tableau `@tab` se note `$tab[i]`. Par exemple avec les déclarations ci-dessus `$chiffres[1]` représente 2.

Affectation.- On peut affecter un tableau à un autre tableau :

```
@digit = @chiffres;
```

le signe d'affectation étant '=', comme en langage C.

On peut même en profiter pour concaténer des tableaux :

```
@caracteres = (@chiffres, '_', @alphabet);
```

Longueur d'un tableau.- L'opérateur '$' permet d'obtenir le dernier indice d'un tableau et donc la longueur moins un de celui-ci. Ainsi `$@chiffres` donne 9.

5.3.2 Chaînes de caractères

5.3.2.1 Séquences d'échappement

Outre les séquences d'échappement du langage C (à savoir `\n`, `\r`, `\t`, `\f`, `\b`, `\v`, `\a`, `\\`, `\'`, `\177`, ou n'importe quelle autre valeur octale, et `\x7f`, ou n'importe quelle autre valeur hexadécimale), on peut utiliser les séquences d'échappement suivantes dans les chaînes de caractères entre guillemets :

Séquence	Signification
\e	Échappement
\C	Tout caractère de contrôle, comme C
\l	Met la lettre suivante en minuscule
\L	Met toutes les lettres suivantes en minuscule jusqu'à \E
\u	Met la lettre suivante en majuscule
\U	Met toutes les lettres suivantes en majuscule jusqu'à \E
\E	Annule l'effet de \L ou \U

5.3.2.2 Fonctions sur les chaînes de caractères

`split(motif, ch)` sépare la chaîne de caractères `ch` en plusieurs éléments, le séparateur étant `motif` et renvoie le tableau constitué de ces éléments.

Par exemple :

```
@t = split(' / ', 'amande / fraise / cerise');
```

renvoie comme tableau `@t` la valeur `('amande','fraise', 'cerise')`.

5.3.3 Tableaux indicés ou associatifs

En Perl, les tableaux associatifs permettent d'obtenir des structures de données dynamiques. Les éléments d'un tableau sont indicés par des entiers commençant à 0 alors qu'un **tableau associatif** est indicé par des valeurs scalaires quelconques, appelées **clés**.

Nom.- Le nom d'un tableau associatif est un identificateur précédé du caractère '%'.

Constante.- Un tableau associatif est une liste de couples de valeurs séparées par des virgules et encadrées par des parenthèses, chaque couple tant de la forme $a => b$:

```
%prix = ('amande' => 30, 'fraise' => 9, 'cerise' => 25);
```

ce qui peut aussi s'écrire, pour simplifier :

```
%prix = (amande => 30, fraise => 9, cerise => 25);
```

Référence à un élément.- L'ordonnée correspondante à l'élément du tableau associatif `%tab` dont l'abscisse est `a` se note `$tab{i}`. Par exemple, avec les déclarations ci-dessus, `$prix{'fraise'}` ou `$prix{fraise}` représente 9.

Structure de contrôle `each`.-

5.4 Historique

Le langage Perl a été conçu pour simplifier l'administration d'un réseau de taille importante. Depuis quelques temps, Larry WALL souhaitait développer un meilleur moyen de générer des rapports de débogage. Il travaillait sous le système d'exploitation UNIX. Il commença par tester l'utilitaire UNIX `awk` mais n'obtint pas les résultats espérés. En s'appuyant sur cette application, il développa la première version de Perl. Ce nouvel outil se révéla beaucoup plus efficace qu'il ne l'avait imaginé. Aussi fit-il cadeau de ce nouveaul langage à quelques programmeurs UNIX. Ces dernier l'enrichirent pour qu'il effectue plus de tâches.