*<https://perl.developpez.com/tutoriels/perl/debuter>*

1. **Variables scalaires :**

* **Define « scalaire » :** *Grandeur qui est suffisamment définie par ses propres mesure et unité, par*

*opposition aux gradeurs vectorielles.*

* Perl, comparé aux autres langages de programmation (comme C ou C++) est un langage peu typé. Il utilise des variables scalaires pour stocker nombres, booleens ou chaines de caractère(s) ; mais aussi des tableaux et tables de hachages.
* Chaque variable commence par un sigil. Pour les variables scalaires il s’agit de « $ ». On trouve le symbole « @ » pour les tableaux et enfin celui de « % » pour les tables de hachages.
* **Define « sigil » :** *Symbole graphique non-alphanumérique représentant une intention. En perl, le sigil*

*sert d’indicateur, notamment pour l’interprétateur*

1. **Listes et tableaux :**

En perl, les tableaux et les listes simples ne sont pas foncièrement différents… Ils regroupent tous les deux des éléments, pouvant même être hétérogènes (nombres, booléens, etc).

**Exemple :** *@tableau = (1, 2, 3, "texte", 4, true, 5, "fin") ;*

1. **Table de hachages :**

Une table de hachages, ou encore liste associative, correspond en réalité à un simple tableau possédant un couple *« clef -> valeur »* pour chacun de ses éléments. De par cette particularité, les tables de hachages, comparées aux tableaux, sont non ordonnées.

**/!\ Attention :** Le sigil pour la table de hachage est « % » mais lorsque l’on veut accéder à une valeur il faut utiliser le sigil « $ ». En effet on accède à la valeur scalaire de l’un des éléments

1. **La condition « unless » :**

La condition « unless » s’utilise de la même manière que la condition « if » mais fonctionne en sens inverse. Elle peut se traduire par « à moins que (condition) soit vérifiée ».

1. **Les boucles « until » et « foreach » :**

* La boucle « until » s’utilise de la même manière que la boucle « while » à la différence qu’elle se réalise tant que la condition reste fausse.
* La boucle « foreach », s’utilise de la même manière qu’une boucle « for » classique mais s’utilise plutôt, elle, sur des tableaux ou des listes associatives.

1. **Variables spéciales (ou réservées) :**

* Comme « <> » correspond à l’entrée par défaut (clavier), il existe aussi en Perl une variable scalaire par défaut de nombreuses fonctions, notée « $\_ ». Elle contiendra également, dans une boucle, le résultat de la ligne en cours.
* La variable « $! » enregistre, elle, l’état de la dernière erreur en cours.
* Le tableau « @ARGV » contient tous les arguments passé à votre programme lors de son appel.
* La liste « %ENV » contient, elle, toutes les informations liées au système sur lequel tourne le programme.

1. **Ecrire des fonctions :**

* En perl les fonctions sont déclarées de la manière suivante : « sub functionName {} ».
* Les paramètres transmis à la fonction sont tous stockés dans la variable tableau spéciale « @\_ ».

1. **Expressions régulières (ou regex) :**

* Les regex forment en quelque sorte un langage très rapide et très puissant, à la base des moteurs de recherche actuels et dédié à la recherche dans des chaines de caractère. Le principe même des expressions régulières consiste simplement à rechercher et/ou remplacer, en général de manière très précise, des éléments d’une chaine de caractère. C’est ce qu’on appelle la correspondance ou la substitution selon le cas.
* Il existe deux syntaxes de regex : POSIX et PCRE. Perl utilise PCRE qui est un peu plus complexe mais beaucoup plus rapide que POSIX (notamment utilisé en PHP par exemple).
* L’opérateur de liaison pour comparer une valeur à une expression régulière est : « =~ ».
* Les expressions régulières en Perl sont déclarées entre deux slashs : « /regex/ ».