Perl TD3 : Fichiers - Fonctions

Résumé sur les fichiers

Un descripteur (ou *handle*) de fichier est le nom d'une connexion d'E/S (Entrée/Sortie) établie entre un programme Perl actif et le monde extérieur.

Perl fournit 3 descripteurs de fichiers qui s'ouvrent chacun automatiquement sur une E/S précise :

— STDIN : entrée standard,
— STDOUT : sortie standard,
— STDERR : sortie d'erreurs standard.

Pour ouvrir tout autre descripteur de fichier, utiliser la commande open() :

```
open (FICDESC, "nom_fichier"); # ouvrir en lecture
open (FICDESC, ">nom_fichier") # ouvrir en ecriture
open (FICDESC, ">>nom_fichier") # ouvrir en ajout
```

Pour fermer un descripteur de fichier :

```
close (FICDESC);
```

L'utilisation est trés semblable à <>.
Ex :

```
open (EP, "/etc/passwd");
open (RES, ">res.txt");
while (<EP>)
   chomp; # supprime le newline à la fin de $_
   print RES "I saw $_ in passwd file.";
close (EP);
close(RES);
```

L'opérateur **die()** prend une liste optionnelle de paramètres, l'affiche (à la façon de **print**) sur STDERR et termine le programme Perl.
Ex :

```
open (DATA, "rapport.embl") || die "Le fichier n'existe pas.";
OU
open (DATA, "rapport.embl") || die "ouverture impossible: $!";
```

(La variable $! contient la chaine d'erreur décrivant l'erreur système la plus récente.)

Fonctions

Un très bref résumé pour savoit définir et appeler une fonction :

```
sub say_hello {  # definition
print "Hello world!\n";
}
...
say_hello();  # appel
```

Possibilité de valeur de retour :

```
sub sum_nb {  # definition
my $a = 6;   # variables locales
my $b = 9;
return $a + $b;  # retourne la valeur 15
}
...
$val = sum_nb();   # appel
```

Possibilité de transmettre des arguments à une fonction. Les arguments d'une fonction sont passés par valeurs et récupérés dans la variable tableau spéciale `@_` .

```
my $a = "world";
sub say {   # definition
print "$_[0] $_[1]!\n";
}
...
say("hello", $a); # appel
say("goobye", "cruel ".$a);
```

Exercice 1

Considérez un fichier **source_file** contenant un texte quelconque. Ecrivez un script Perl qui recopie ce fichier dans un fichier **result_file** de sorte que chaque ligne soit précédée de son numéro de ligne.

Exercice 2

1. Definissez une fonction perl **moyenne**, qui calcule la moyenne de ses arguments, quelque soit leur nombre.
2. Definissez une fonction perl **printHash**, qui prend en argument une table de hachage et l'affiche sur la sortie standard, un couple clé/valeur par ligne.
3. Definissez une fonction perl **replaceTab**, qui prend en argument deux scalaires et un tableau, et qui remplace dans le tableau toutes les occurrences du premier scalaire par le second. Le tableau passé en argument devra être modifié.

Exercice 3

Reprenez le fichier **kinases_map.txt** du TD précédent.

1. Ecrivez un script Perl qui lit ce fichier et affiche, séparés par des tabulations, le "gene symbol", le "gene name" et le "cytogenetic location".
2. Modifiez votre script pour qu'il affiche seulement les "tyrosine kinase genes" (i.e. ceux dont le "gene name" contient le mot tyrosine).
3. Modifiez votre script pour afficher seulement les gènes localisés sur le chromosome 9.
4. Modifiez votre script pour afficher seulement les gènes entrés dans la base de données après 1995.
5. Modifiez votre script pour afficher seulement les gènes vérifiant tous les critères précédents simultanément.

Exercice 4

Modifiez le script de l'exercice précédent pour proposer un menu à l'utilisateur, lui permettant de choisir entre les différents affichages proposés