ISMAIL Saman Groupe : IE1-02

Compte rendu projet ScIn1B

Exercice 1 : Exo1.pl

Un programme Perl qui, à partir d'un texte saisi, reconnait un mot se terminant par la lettre "a" et l'enregistre dans $1. Affiche ensuite le texte précédant le mot reconnu, le mot reconnu entre "<" et ">" et le reste du texte saisi. Affiche ensuite soit le contenu de $1, soit un message si rien n'a été trouvé.

```
{
    #On affiche ce qu'on demande de l'utilisateur
    print ("Veuillez saisir votre texte - terminez votre saisie en appuyant sur ENTREE - quittez le programme avec Ctrl + z\n>");

    my $texte = <STDIN>;  #On attend que l'utilisateur rentre le texte et appuie sur ENTREE

    chomp ($texte); #On supprime \n car l'utilisteur a tapé sur entrée
```

La fonction print() nous permet d'afficher un texte sur l'écran.

<STDIN> nous permet de lire sur l'entrée standard c'est-à-dire le clavier et dés que l'utilisateur tape quelque chose ou pas puis appuie sur entrée pour qu'on récupère la valeur dans la variable $texte

La fonction chomp() est une fonction intégrée dans perl qui nous permet de supprimer le dernier symbole \n car l'utilisateur a tapé sur entrée juste avant avec <STDIN>.

```
if($texte =~ s/ (\w+a) / <$1> /)
{
    print("Mot reconnu: |$texte|\n");
    print('$1'." contient \'$1\'");
}
```

Ce code veut dire que si le texte entré par l'utilisateur contient un espace suivi d'un mot finissant par un a suivi d'un espace alors on entoure le mot finissant par un a avec < et > puis on affiche la phrase entière entrée par l'utilisateur en la mettant entre deux | et afficher le premier mot finissant par un a dans la phrase.

```
elsif($texte =~ s/ (a) / <$1> /)
{
    print("Mot reconnu: |$texte|\n");
    print('$1'." contient \'$1\'");
}
```

Sinon si le texte entré par l'utilisateur contient un espace suivi d'un a suivi d'un espace alors on entoure a avec < et > puis on affiche la phrase entière entrée par l'utilisateur en la mettant entre deux | puis afficher le premier a dans la phrase.

```
elsif($texte =~ s/(\w+a$)/<$1>/)
{
    print("Mot reconnu: |$texte|\n");
    print('$1'." contient \'$1\'");
}
```

Sinon si le texte entré par l'utilisateur contient un espace suivi d'un mot finissant par un a en fin de ligne alors on entoure le mot finissant par un a en fin de ligne avec < et > puis on affiche la phrase entière entrée par l'utilisateur en la mettant entre deux | et afficher le mot finissant par un a en fin de ligne.

```
else
{
    print("Aucun mot reconnu: | |");
}
```

Sinon on affiche sur l'écran Qu'aucun mot finissant par un a n'a été trouvé dans la phrase entrée par l'utilisateur.

Fin Exercice 1

Exercice 2 : Exo2.pl

Un programme Perl qui copie un fichier puis en modifie le contenu. Lors de la modification, chaque occurrence de la chaîne "Fred" (insensible à la casse) sera remplacée par la chaîne "Larry". Le nom du fichier à copier est fourni en tant qu'argument. Le nom du fichier copié et modifié est le même que celui du fichier d'origine et il on lui ajoute l'extension ".out". Si le fichier d'origine ne contient pas d'extension, on ajoute ".out". Si le fichier d'origine contient une extension, alors celle-ci est remplacée par ".out". On affiche le nom du fichier généré et le nombre de remplacements effectués.

```
my $fich = $ARGV[0];  # prendre le premier argument
my $lec;       #le fichier en lecture
my $ligne;     #la ligne qui est en train d'être lue
my $op;        #le output en ouverture
my $compteur=0; # le compteur des changements éfectués
```

Dans cette partie on déclare les variables qui nous serons nécessaires par la suite.

L'utilité de chaque variable est précisée dans le code.

```
open ( $lec, "<", $fich ) or die "erreur sur $fich";

if($fich =~ /(.+)\.(.+)/)
{
$fich=$1; }
```

On essaye d'ouvrir le fichier passé en argument en lecture. S'il ne s'ouvre pas, on affiche erreur sur + nom de fichier et le programme s'arrête.

Si le fichier passé en argument est sous la forme toto.tata c'est-à-dire avec une extension alors on lui enlève l'extension en prenant le premier groupe de capture.

```
 my $outPut= "$fich.out";
open( $op , ">",$outPut  ) or die "erreur sur $outPut";
```

On ajoute l'extension .out au nom du fichier et ou l'ouvre en écriture avec >.

```
while ( <$lec> )
    {
      $ligne=$_;

      if ( $ligne =~ s/Fred/Larry/gi)
        {
          print $op "$ligne\n";
        }

      $compteur = $compteur + 1;

    }

    print("Fichier généré: '$outPut'\n");
    print("Remplacement effectuées: '$compteur'\n");

close($lec);
```

On créer une boucle while pour lire ligne par ligne le fichier passé en argument.

On regarde ligne par ligne s'il y a « Fred » sans espaces autour sur toute la ligne en ne pas respectant la casse c'est-à-dire les majuscules et minuscules et on le remplace pas « Larry » en respectant la casse et on ajoute cette ligne à au fichier qui a pour extension .out.

On augmente le compteur de 1 à chaque fois qu'on trouve un 'Fred'.

Remarque importante : Dans le sujet du projet on doit trouver 11 changements sauf que le fichier Exo2.txt contient 12 fois le mot 'Fred' donc il y a bien 12 changements mais le compteur nous donne 11 changements.

Puis on affiche sur l'écran le fichier généré et le nombre de changements effectués

Et enfin on ferme le fichier qui a été ouvert en lecture avec la fonction close().

Fin Exercice 2

Exercice 3 : Exo3.pl

Un programme Perl qui va demander à l'utilisateur de deviner un nombre entre 1 et 100. Tant que l'utilisateur n'a pas deviné le nombre correct, le programme continue de lui demander de saisir un nombre. Pour chaque nombre saisi, le programme affiche "Trop petit" ou "Trop grand" selon le nombre à deviner. Si l'utilisateur saisit "quitter" (insensible à la casse), ou une ligne vide, le programme doit quitter et afficher un message à l'utilisateur. Si l'utilisateur devine, le programme affiche un message à l'utilisateur et quitte.

```
my $reponse=-1; # la réponse de l'utilisateur -1 quand l'utilisateur n'a pas encore mis de reponse
my $random = int(rand(99))+1;
```

Déclaration et initialisation des variables $reponse et $random.

$reponse est initialisée à -1 jusqu'à ce que l'utilisateur l'initialise lui-même.

$rendom est comprise 1 et 100, on a utiliser la fonction rand pour trouver un nombre aléatoire et int() pour enlever la virgule.

```
while ( int($reponse) != $random)
{
    print ("Veuillez saisir un nombre entre 1 et 100:: "); #On affiche ce qu'on demande de l'utilisateur


    $reponse = <STDIN>;  #On attend que l'utilisateur rentre valeur entre 1 et 100
    chomp ($reponse); # on enléve le \n de la réponse
```

On crée une boucle while pour dire que tant que la réponse entrée par l'utilisateur est différente du nombre aléatoire initialisé juste avant : On demande à l'utilisateur de rentrer un valeur entre 1 et 100,

On récupère cette valeur dans la variable $reponse avec <STDIN> et on lui enlève \n (saut de ligne) avec la fonction chmod().

```
last if("$reponse" =~ /quitter/i);

last if("$reponse" eq "");
```

On quitte le programme avec last si la $reponse de l'utilisateur est 'quitter ' en ne pas respectant la casse (i permet de faire cela).

On quitte le programme avec last si la $reponse de l'utilisateur est une chaine vide.

```
if($reponse < $random && $reponse >= 1 )
{
    print("Trop petit. Veuillez réessayer !\n");
}
```

Si la réponse entrée pas l'utilisateur est inférieure au nombre aléatoire et supérieur ou égal à 1 alors on affiche à l'écran que la valeur est trop petite.

```
elsif($reponse > $random && $reponse <=100)
{
    print("Trop grand. Veuillez réessayer !\n");
}
```

Sinon si réponse entrée pas l'utilisateur est supérieure au nombre aléatoire et inférieure ou égale à 100 alors on affiche à l'écran que la valeur est trop grande.

```
elsif($reponse < 1 ||$reponse > 100 )
{
    print("Vous avez mis un nombre hors l'intervalle entre 1 et 100 !\n");
}
```

Sinon si réponse entrée pas l'utilisateur est inférieure à 0 ou supérieure à 100 alors on affiche à l'écran que la valeur entrée est hors l'intervalle demandé.

```
elsif($reponse == $random)
{
    print("Bravo vous avez trouvé !\n");
}
```

Sinon si réponse entrée pas l'utilisateur est égale au nombre aléatoire alors on affiche que le joueur a gagné.

On pourrait éventuellement ajouter le nombre de tentatives que l'utilisateur a eu besoin pour gagner mais ce n'est pas demandé dans le sujet.

Fin Exercice 3

Exercice 4 : Exo4.pl

Un programme Perl qui, pour un nombre saisi par l'utilisateur, affiche tous ses diviseurs (sauf 1 et le nombre lui-même). Par exemple, pour le nombre 99, le programme affiche les diviseurs 3, 9, 11, 33. Si le nombre n'a pas de diviseurs, le programme affiche qu'il est premier. Si l'utilisateur ne saisit pas un nombre, le programme l'indique à l'utilisateur, sans essayer de déterminer des diviseurs.

```
my $nb = $ARGV[0];
my @div;
```

Déclaration des variables $nb et @div

$nb est le nombre passé en argument on l'initialise avec le premier élément de la liste des arguments.

@div est une liste des diviseur du nombre qui sera initialisée plus tard.

```
if($#ARGV == 0 && $nb =~ /^\d+$/)
{

    print("Vérification du nombre <$nb>\n");
    @div = diviseurs($nb);
    if($#div == -1)
    {
        print("$nb est un nombre premier");
    }
    else
    {
        print "$nb est divisible par ";
        for (my $i=0; $i <= $#div; $i++)
        {
            print "$div[$i] ";
        }
    }

}
```

Si le nombre d'éléments dans la liste des arguments est 1 et que le premier paramètre est un nombre ^ pour début de ligne \d+ pour des chiffres et $ pour la fin de ligne.

Alors on affiche qu'on vérifie le nombre passé en paramètre et on cherche les diviseurs de ce nombre grâce à la fonction diviseurs($nb) et on les stock dans la liste @div.

Si la liste @div ne contient aucun élément alors on affiche que c'est un nombre premier sinon on crée une boucle pour afficher les diviseurs un par un de la liste @div.

```
else
{
    print("Vérification du nombre <$nb>\n");
    print("Vous n'avez pas saisi un nombre");

}
```

Si le nombre d'éléments dans la liste des arguments est différent de 1 et que le premier paramètre n'est pas un nombre dans ce cas on affiche sur l'écran en disant que l'utilisateur n'a pas entré un nombre.

```
sub diviseurs
{
    my $nb = shift;
    my @diviseurs = ();

    foreach my $diviseur (2 .. ($nb/2))
    {
        push @diviseurs, $diviseur unless $nb % $diviseur;
    }
    return @diviseurs;
}
```

La fonction donnée dans le sujet qui nous a permis de mettre tous les diviseurs d'un nombre dans une liste.

Fin Exercice 4