Proposition Solution Epreuve option D

Concours de Contrôleur de la Concurrence, de la Consommation et de la Répression des Fraudes des 6 et 7 mars 2017.

RKL XVI

Proposition du 25 janvier 2018

A. Le point le plus proche

Les fonctions des questions A.1 et A.2 sont rédigées dans le même script de fichier. La formule de la distance euclidienne entre deux points est égale à : $\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$.

```php
<?php
  //A.1 Fonction calculant le point le plus proche
  function le_plus_proche($xP, $yP, $x, $y, $N){

    $distance_proche_trouve = null;
    $resultat = null;

    //Parcour les tableaux et calcul le point le plus proche
    for($i = 0 ; $i < $N ; $i++){

      // Calcul la distance euclidienne d'un point d'indice $i des tableaux $x et $y
      $distance_evalue = distance($xP, $yP, $x[$i], $y[$i]);

      // Si la nouvelle distance evaluee est plus courte que son antecedent alors elle devient la distance
      // la plus courte.
      // La variable $resultat stocke alors la position de la variable $i (l'indice des tableaux).

      if ($distance_proche_trouve == null || $distance_evalue < distance_proche_trouve){

        $distance_proche_trouve = $distance_evalue;
        $resultat = $i;

      }
    }

    return $resultat;
  }

  //A.2 Fonction calculant la distance euclidienne
  function distance($xA, $yA, $xB, $yB){

    // En php ** 2 permet d'elever le nombre au carre
    return sqrt(($xB - $xA)** 2 + ($yB - $yA)** 2);

  }

?>
```

B. Chiffres, lettres et caractères de ponctuation

B.1 Compter

```php
<?php
  function compter($s, &$cpt_chiffre, &$cpt_lettre,  &$cpt_ponctuation){

    // Parcourir caractere apres caractere la chaine $s
    for($c = 0 ; $c < strlen($s) ; $c++){

      // Utilisation des expressions regulieres pour identifier les chiffres , lettres et ponctuation
      if ( preg_match("/[[:digit:]]/" , $s[$c]) ){ // S'il s'agit d'un caractere numerique

        $cpt_chiffre ++;

      } elseif ( preg_match("/[[:alpha:]]/" , $s[$c]) ){ // S'il s'agit d'une lettre

        $cpt_lettre ++;

      } elseif ( preg_match("/[[:punct:]]/" , $s[$c]) ){ // S'il s'agit d'un caractere de punctuation

        $cpt_ponctuation++;

      }
    }
  }
?>
```

B.2 Fonction main

Prérequis : Cette fonction s'écrit dans le même script que la fonction compter vue à l'exercice B.1

```php
<?php

  function main(){

    $nbre_chiffre  = 0;
    $nbre_lettre  = 0;
    $nbre_punct = 0;
    $chaine = "Vive la programmation !!! Et 1, et 2, et 3 zero!";

    // Appel de la fonction compter
    compter($chaine, $nbre_chiffre , $nbre_lettre , $nbre_punct);

    // La fonction compter stocke les resultats dans les variables $nbre_chiffre , $nbre_lettre et
    // $nbre_punct. Cela est rendu possible car ces variables sont des alias (&) dans la fonction compter.
    // Les modifications realisees dans les variables de la fonction compter affecte
    // les variables de la fonction main. Il est donc possible d'afficher les resultats :
    echo "Nombre de chiffre : $nbre_chiffre".'<br/>';
    echo "Nombre de lettre : $nbre_lettre" .'<br/>';
    echo "Nombre de ponctuation : $nbre_punct".'<br/>';

  }
?>
```

C. Consonnes et voyelles

C.1 Voyelle

```php
<?php

  // Recherche des voyelles par l'utilisation  des expressions regulieres
  function voyelle($c){

    // Retourne VRAI si les voyelles "aeiou" et "y" (lettres  en minuscules ou en majuscules)
    // sont trouvees. FAUX dans le cas contraire.
    return preg_match("/[aeiouyAEIOUY]/" , $c);

  }

?>
```

C.2 Compter_lettres

Prérequis : La fonction compter_lettres se trouve dans le même script que la fonction voyelle codée à l'exercice C.1

```php
<?php

  function compter_lettres($s , &$cpt_voyelle , &$cpt_consonne , &$cpt_autres){

    //Parcourir la chaine $s caractere apres caractere
    for($c = 0 ; $c < strlen($s) ; $c++){

      // Si l'on trouve un caractere alphabetique, distinguer s'il s'agit
      // d'une voyelle ou d'une consonne.

      if( preg_match("/[[:alpha:]]/" , $s[$c]) ){

        if(  voyelle($s[$c]) ) $cpt_voyelle++;
        if( !voyelle($s[$c]) ) $cpt_consonne++;

       } else { // Sinon, il s'agit d'un autre caractere

        $cpt_autres++;

       }
     }
  }
?>
```

D. Nombres impairs

Prérequis : L'utilisateur donne au clavier le nombre voulu dans un formulaire php (balise <form >). Le programme récupère le nombre de l'utilisateur à l'aide la variable globale $_POST.

```php
<?php

  $n = $_POST["nombre_utilisateur"];
  $compteur =0;
  $iteration = 1;

  while($compteur <= $n){

    // Pour tout entier naturel appel iteration , iteration % 2 == 0 est un nombre pair.
    if ($iteration % 2 != 0) {// Si nous trouvons un nombre impair, afficher le rsultat

      echo "$i ";
      $compteur++;

    }

    // Comme nous trouvons un nombre pair, passons au nombre suivant.
    $iteration ++;

  }
?>
```

E. Positiver les tableaux

```php
<?php
  function(&$a, $N){

    // Positiver un nombre negatif consiste a trouver sa valeur absolue ou a le multiplier par - 1
    for($i = 0 ; $i < $N ; $i++)
        $a[$i] = abs($a[$i]);

  }
?>
```

F. Tracabilité de l'huile d'olive

7.a

```sql
SELECT p.nom
FROM Producteur p
INNER JOIN Groupement g ON g.code =p.codeGroupement
WHERE g.nom ='BioOliCenter';
```

7.b

```sql
SELECT p.code, SUM(r.qteRecoltee)
FROM Producteur p
INNER JOIN Parcelle parc ON parc.codeProducteur =p.code
INNER JOIN Recolte r ON r.CodeParcelle =parc.code
GROUP BY p.code;
```

7.c

```
1 SELECT parc.code
2 FROM Parcelle parc
3 WHERE parc.code NOT IN (SELECT tp.codeParcelle
4                         FROM TraitementPhytosanitaire tp
5                         WHERE tp.date >= '2011-01-01');
```