Sujet 1 complémentaire
Buts : Types, Fonctions, Tableaux

Avant de commencer

Vous y stockerez dans le répertoire `TP1` le script *tp1Complementaire.php* l'ensemble des développements produits tout au long de ce TP complémentaire. N'oubliez pas de mettre à jour systématiquement votre dépôt git $POO - TP1$.

Figures

Question 1.
Définissez la fonction `dessinerContourRectangle` qui prend en paramètre un motif (caractère), le nombre de lignes et le nombre de colonnes du rectangle. Cette fonction affichera le contour d'un rectangle de dimension $lignes \times colonnes$. Ce contour sera composé de motifs séparés par des espaces.

Question 2.
Définissez la fonction `dessinerTriangleRectanglePlein` qui prend en paramètre un motif (caractère), le nombre de caractères dont sont composés les cotés définissant l'angle droit du triangle. Cette fonction affichera le triangle composé de motifs séparés par des espaces. Le rendu souhaité pour un nombre de caractères égal à 5 et le motif `*` est :

```
        *
      * *
    * * *
  * * * *
* * * * *
```

Question 3.
Définissez la fonction `dessinerTriangleIsocelePlein` qui prend en paramètre un motif (caractère), le nombre de caractères dont est composée la base du triangle. Cette fonction affichera le triangle composé de motifs séparés par des espaces. Attentions aux triangles dont la base est de taille paire. Le rendu souhaité pour une base de taille 4 et 5 et le motif `*` est :

```
   *
  * *
* * * *

    *
  * * *
* * * * *
```

Le crible d'Eratosthène

L'objectif du crible d'Eratosthène[1] est de déterminer les nombres premiers inférieurs à un nombre entier n. Pour ceci on utilise le crible d'Eratosthène. Le crible procède de la manière suivante :

- On utilise un tableau de $n+1$ éléments. (1)
- On raye les cases 0 et 1 (ces nombres ne sont pas premiers). (2)
- On passe ensuite à 2 qui n'est pas rayé : c'est un nombre premier. En sautant de 2 en 2, on atteint les multiples de 2. On les raye. (3)
- On passe ensuite à 3 qui n'est pas rayé : c'est un nombre premier. En sautant de 3 en 3, on atteint les multiples de 3. On les raye. (4)
- On passe ensuite à 4 qui est rayé : ce n'est pas un nombre premier. (5)
- On passe ensuite à 5 qui n'est pas rayé : c'est un nombre premier. En sautant de 5 en 5, on atteint les multiples de 5. On les raye. (6)
- ...

Voici le début du déroulement de l'algorithme :

(1) :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
(2) :			2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
(3) :			2	3		5		7		9		11		13		15		17		19	
(4) :			2	3		5		7				11		13				17		19	
(5) :			2	3		5		7				11		13				17		19	
(6) :			2	3		5		7				11		13				17		19	
	...																				

Question 4.
Trouvez à la main tous les nombres premiers inférieurs à 50.

Question 5.
Définissez une fonction `cribleEratosthene` qui permet de déterminer les nombres premiers inférieurs à un nombre entier n passé en paramètres. Cette fonction retournera le tableau des nombres premiers.

Indications :
On se propose ici de travailler avec un tableau de $n+1$ éléments où les indices du tableau seront assimilés aux nombres recherchés. Les cases du tableau seront initialisées à *true*, c'est à dire que tous les nombres sont supposés être premiers avant d'appliquer l'algorithme. Au fur est à mesure du déroulement de l'algorithme, les cases ne correspondant pas aux nombres premiers seront mises à $false$.

Question 6.

[1] Astronome, géographe mathématicien et philosophe grec, il séjourna longtemps à Athènes avant d'être appelé en Alexandrie vers -245 par le pharaon Ptolémée qui lui confia l'éducation de son fils, puis la direction de la fameuse bibliothèque d'Alexandrie. Ses contributions furent variées : philosophie, grammaire, chronologie, etc... Mais il est surtout connu comme mathématicien grâce au célèbre crible, méthode pour trouver les nombres premiers. Il fut le premier à évaluer de façon exacte la longueur de la circonférence terrestre.

On peut diminuer le coût de cet algorithme en remarquant qu'il est inutile de le poursuivre au-delà de $\sqrt{n}$. *Modifiez* en conséquence la fonction préalablement proposée.

Les carrés des chiffres

Jeu :
- On prend un entier positif n quelconque.
- On additionne les carrés de ses chiffres : on obtient un nouveau nombre.
- On recommence : on additionne les carrés des chiffres du résultat - on trouve un nouveau nombre.
- On recommence : $\cdots$

Exemple : $n = 45$

Etape	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	$\cdots$
n	45	41	17	50	25	29	85	**89**	145	42	20	4	16	37	58	**89**	145	42	$\cdots$

Le but de cet exercice est de créer un programme de façon à aborder de manière expérimentale l'apparition de séquences périodes dans les suites composées de carrés des chiffres des nombres qui la composent.

Question 7.
Définissez la fonction `afficheChiffres` qui, connaissant un nombre entier n passé en paramètre, affiche les chiffres du nombre. L'ordre d'affichage n'a pas d'importance.

Question 8.
Définissez la fonction `sommeCarresChiffres` qui, connaissant un nombre entier n passé en paramètre, retourne la somme des carrés des chiffres du nombre.

Question 9.
On cherche maintenant à détecter les suites périodiques qu'on voit apparaître lorsqu'on applique cette opération à un nombre quelconque, de façon récurrente.

Définissez la fonction `determinePeriode` permettant de détecter de telles séquences périodiques. Cette méthode prend en paramètre un nombre n et retourne un tableau contenant la séquence détectée.

Question 10.
Il semble que les périodes possibles soient en nombre très limité.

Définissez la méthode `listePeriodes` qui prend en paramètre une limite maximale et retourne un tableau de séquences périodiques obtenues par les nombres compris entre 1 et cette limite incluse. Affichez les séquences périodiques obtenues.

Question 11.
Optimisez la méthode précédente afin qu'une séquence périodique ne soit pas ajoutée à la liste si elle y est déjà présente. On pourra pour ceci définir une méthode `estPresente` qui prend en paramètre la première valeur d'une séquence ainsi qu'un tableau de séquences périodiques et

qui vérifie si cette valeur est déjà présente dans l'une des séquences du tableau. Cette méthode retournera vrai si c'est le cas, faux sinon.

Affichez à nouveau les périodes obtenues pour des nombres compris entre 1 et 10000.