Tant que ceci, faire cela

7 décembre 2008

Résumé des épisodes précédents

Dans les épisodes précédents, nos fabuleux explorateurs de l'informatique (vous!) ont découvert les boucles `for` et ont été émerveillés par les possibilités infinies de faire faire les tâches répétitives à leur machine plutôt que de s'user les doigts à tapoter au clavier. En les combinant au pouvoirs des tests conditionnels `if else end`, ils ont réussi à vaincre de nombreux exercices, mais aujourd'hui, de nombreux défis grecs les attendent. Guidés par le sage EN'CA DREN, ils partent à l'aventure.

La préparation

Avant de partir, nos héros ont appris une nouvelle astuce : les boucles `while`.
Résumé :

Syntaxe

```
while condition
  body
end
```

Description une boucle `while` exécute le corps de la boucle `body` tant que la condition `condition` est vraie.

Exemple

```
petit=2 // une petite valeur
grand=1234 // une grande valeur
while petit < grand // notre condition
  petit = 2*petit // on double la taille du petit
  // plus efficace que la soupe !
end
// on ne sort de la boucle que quand petit est devenu plus grand que grand
```

1 Niveau -1 : “max payne”

Vous aviez déjà triomphé de l'algorithme du maximum il y a quelques semaines, mais il est de retour. Utilisez votre nouvelle arme pour l'achever!

Aide: Votre nouvelle arme, c'est le `while`

1. Créer un tableau `tab` de 100 éléments aléatoires;

2. Créer un entier `index` initialisé à 1. On s'en servira pour savoir sur quelle colonne du tableau on est ;
3. Créer un entier `index_max` initialisé à 1. On s'en servira pour se souvenir dans quelle colonne se trouve le maximum actuel ;
4. Appliquer la séquence suivante :
 - (a) regarder si l'élément de `tab` dans la colonne `index` est strictement plus grand que l'élément dans la colonne `index_max` ;
 - (b) si oui, `index` devient le nouvel `index_max` ;
 - (c) si non, on ne fait rien ;
 - (d) incrémenter `index` de 1 ;
 - (e) continuer **tant que** `index` est inférieur (ou égal ?) à 100 ;
 - (f) afficher l'élément de `tab` à la colonne `index_max`.

2 Niveau 0 : La disparition

Au détour d'un chemin, vous rencontrez un écrivain qui a décidé d'écrire un livre ... sans lettre 'e'. Il vous demande de vérifier son livre et ses 10000 mots ! Heureusement, il y a `SCILAB`.

1. Créer un tableau `mot` qui contient ... un mot de la langue française (une lettre par colonne) ;
2. Créer un entier `index` valant 1. Ce sera notre indice de départ ;
3. Appliquer la séquence suivante :
 - (a) **tant que** `index` est plus petit (ou égal ?) que la taille du tableau, et que la lettre de `mot` dans la colonne `index` n'est pas 'e' ;
 - (b) incrémenter `index` de 1 ;
 - (c) une fois sorti de la boucle, tester la valeur de `index`
 - i. si elle est égale à la taille du tableau `mot` (+ 1 ?), afficher "ok !" ;
 - ii. sinon, afficher "lettre 'e' trouvée dans la colonne `index`".

Pour information, un tel livre existe et s'intitule justement "la disparition"

3 Niveau 1 : Le lapin crétin

Voici votre premier vrai défi, posé par un animal pas très malin. On lui a posé une question et vous devez l'aider !

"Mon adversaire choisi un nombre entre 3 et 103 inclus. Je dois trouver lequel en un minimum de coups, il me répond par plus grand ou plus petit !"

Vous aller écrire un programme pour aider ce pauvre lapin.

1. Poser `cible` avec une valeur aléatoire entre 3 et 103. Ce sera la valeur à trouver ;
2. Lire un entier depuis la console et le stocker dans `devine` ;
Aide: la fonction `input` est pratique dans ce cas
3. Suivant la valeur de `devine`, afficher un petit message qui indique au joueur s'il est plus grand, plus petit ou correct ;
Aide: Cette fois, ce sera `printf` pour s'en sortir
4. Continuer **tant que** le joueur n'a pas trouvé.

4 Niveau 2 : Le crible

Voici votre deuxième défi, posé par le mathématicien grec ERATOSTHÈNE.

“Trouvez les 100 premiers nombres premiers!”

Aide: On rappelle qu’un nombre premier est un nombre qui n’a que deux diviseurs : 1 et lui-même

1. Créer un tableau `nombres_preiers` qui ne contient qu’un seul élément, le nombre premier 2. Ce tableau contiendra tous les nombres premiers trouvés ;
2. Écrire un petit bout de code qui, prenant deux entiers, teste si le plus petit des deux divise l’autre ;
Aide: `pmodulo` calcul le reste de la division euclidienne entre deux nombres
3. Écrire une boucle `for` qui, pour un entier n donné, teste si un des éléments de `nombres_preiers` divise n ;
4. Remplir le tableau `nombres_preiers` de la façon suivante
 - (a) Commencer le test par un entier n qui vaut 2. On va tester sa primalité ;
 - (b) Regarder s’il existe un élément de `nombres_preiers` qui divise n ;
 - (c) Si oui, n n’est pas premier, on ne fait rien ;
 - (d) Si non, n est premier, on l’ajoute à la fin de `nombres_preiers` ;
 - (e) Incrémenter n ;
 - (f) Continuer **tant que** la taille de `nombres_preiers` est plus petite que 100.