Les nombres consistants
Programmation avec Xcas

1ière

Fiche professeur

Auteur : Pierre Lapôtre

But de l'activité : écrire un programme :

Compétences engagées :
notions d'algorithmique :
✓ boucle « pour »
✓ boucle « tant que »
✓ test conditionnel
Pré-requis :
✓ syntaxe de quelques instructions, affectation, partie entière (`floor`)
✓ création d'une liste

Matériel utilisé : Un ordinateur équipé de Xcas
Durée indicative : 1 heure
Nom du logiciel utilisé : Xcas
Documents utiles à télécharger :
9ième Championnat International des Jeux Mathématiques et Logiques :
http ://www.vinc17.org/cijml/c09fi/c09fi1.html

Déroulement de la séance :
La progressivité dans la complexité permet la réalisation d'un programme efficace.
Pour le réaliser on décompose le problème en plusieurs objectifs simples et successifs. On considère les étapes suivantes qui constituent en fait l'algorithme de calcul.

Étape 1 : étant donné un nombre compris entre 100 et 999,
extraire `a` : le chiffre des centaines, `b` : le chiffre des dizaines, `c` : le chiffre des unités.
Ce sera l'occcasion d'utiliser la partie entière d'un nombre.

Étape 2 : multiplier ces trois chiffres.

Ces deux étapes peuvent constituer un petit programme simple réalisable aussi à la calculatrice.

Tant que le résultat est supérieur à 100, recommencer ces deux étapes.

Cette boucle « tant que » s'arrêtera dès que le résultat sera inférieur ou égal à 99.

Étape 3 : lorsque le résultat est inférieur à 100, obtenir `d` : le chiffre des dizaines et `f` : celui des unités.

Étape 4 : multiplier ces deux chiffres.

Tant que le résultat est supérieur à 10, recommencer ces deux étapes.

C'est la répétition de l'étape 1 adaptée aux nombres inférieurs à 100. Lorsque cette deuxième boucle « tant que » s'arrêtera on obtiendra le résultat final de l'algorithme décrit, c'est-à-dire un nombre à un chiffre.
En créant le compteur `k`, dénombrant le nombre d'étapes 2 et 4, on obtient la persistance du nombre considéré (réalisable aussi à la calculatrice).
Une boucle « pour » placée en début de programme permet de calculer la persistance de tous les entiers de 11 à 999. (l'initialisation à 11 est une solution de facilité, il serait intéressant de s'interroger sur l'initialisation la plus grande possible mais la rapidité de calcul du logiciel rend cette recherche superflue).
Enfin un test conditionnel sans alternative permet de ne retenir que les nombres consistants ou superconsistants selon que l'on mette 4 ou 5 dans la condition ligne 13.
Deux listes vides, `L4` et `L5`, ayant été créées au début du programme, elles sont complétées au fur et à mesure de la progression du programme en listes des nombres consistants ou superconsistants selon le cas.

Remarque : cet algorithme suppose que la suite des produits successifs des chiffres est décroissante. Ce résultat pourra être admis. Voici néanmoins une justification :

On considère un entier x dont l'écriture décimale comporte $n+1$ chiffres ($n \geqslant 1$).
x s'écrit donc :
$x = 10^n a_n + 10^{n-1} a_{n-1} + \ldots + 10^2 a_2 + 10 a_1 + a_0$
avec $a_n \in \{1,2,\ldots,9\}$ et $a_i \in \{0,1,2,\ldots,9\}$ pour $i \in \{0,1,\ldots,n-1\}$.
Soit alors la différence :

$$\begin{aligned}\Delta &= 10^n a_n + 10^{n-1} a_{n-1} + \ldots + 10^2 a_2 + 10 a_1 + a_0 - a_n \times a_{n-1} \times \ldots a_1 \times a_0 \\ &= a_n(10^n - a_{n-1} \times \ldots a_1 \times a_0) + 10^{n-1} a_{n-1} + \ldots + 10^2 a_2 + 10 a_1 + a_0\end{aligned}$$

Or, $a_{n-1} \times \ldots a_1 \times a_0 \leqslant 9^n$ donc $\Delta > 0$.
Ceci montre que le produit des chiffres d'un nombre est strictement inférieur à ce nombre. En itérant le procédé, on obtient une suite finie strictement décroissante d'entiers, le dernier terme étant strictement inférieur à 10.

Proposition de programme Xcas

```
consist():={
local k,L4,L5,j,n,a,b,c,d,f;
L4:=[];
L5:=[];
for(j:=11;j<=999;j++) {n:=j;k:=0;
while (n>=100){
k:=k+1;
a:=floor(n/100);b:=floor((n-100*a)/10);c:=n-100*a-10*b;
n:=a*b*c};
while (n>=10){
k:=k+1;
d:=floor(n/10);f:=n-10*d;
n:=d*f};
if (k=4){L4:=append(L4,j)};
if (k=5){L5:=append(L5,j)}};
return L4, size(L4),L5, size(L5)}
:;
```