CALCULATRICE PROGRAMMABLE

(suite du Petit vert n°16)

Objectif des programmes officiels : « Savoir programmer le $n^{ième}$ terme d'une suite définie par une relation de récurrence $u_{n+1} = f(u_n)$ et une condition initiale ».

Exemple : Calculer le $20^{ème}$ terme (u_{19}) de la suite définie par $\begin{cases} u_0 = 2 \\ u_n = 2 \times u_{n-1} - 3 \end{cases}$

Nous avons déjà vu, dans le n°13 du Petit Vert, qu'on pouvait obtenir le résultat en programmant la fonction $u \mapsto 2u - 3$, et en réitérant 19 fois l'appel à cette fonction (c'est à dire en tapent 19 fois de suite sur la touche d'exécution du programme, à partir de la valeur initiale u = 2).

Mais nous désirons obtenir directement u_{19} à l'affichage, en laissant à la machine le soin de "gérer la comptabilité".
Si l'on dispose d'un langage évolué (tel PASCAL), l'écriture du programme correspond à peu près directement à l'écriture mathématique de l'énoncé ci-dessus :

```
FUNCTION U (N:INTEGER) : REAL;
   BEGIN
   IF N=0 THEN U:=2
            ELSE U := 2*U(N-I)-3
   END;
BEGIN
   WRITELN(U(19))
END.
```

Cela signifie que U est une fonction réelle de la variable entière N

Début du "programme principal"
Demande au programme d'écrire u_{19}

Mais la plupart des langages informatiques ne permettent pas une telle écriture (dite "récursive") ; il faut en donner une version "itérative", ce qui implique que l'utilisateur définisse un "compteur" (pour compter le nombre de fois que l'on fait le calcul).
Par la suite, on notera u le terme général de la suite, et i l'indice courant (au départ, = 0 ; à la fin, i = 19).

L'algorithme est le suivant :

$\left.\begin{array}{l} i \leftarrow 0 \\ u \leftarrow 2 \end{array}\right\}$ initalisations

Tant que $i < 19$ faire [$u \leftarrow 2u - 3$; $i \leftarrow i + 1$]

Ecrire u

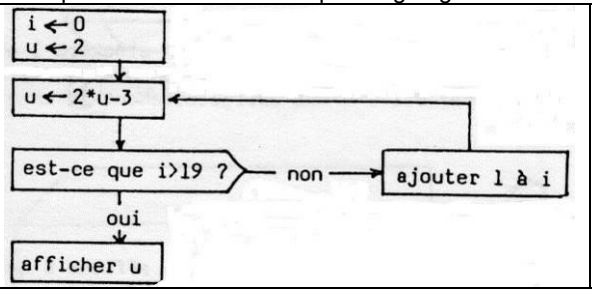
Ce qui, en PASCAL, donne :

```
I:=0 ; U:=2 ;
WHILE I < 19 DO
          BEGIN U:=2*U-3 ; I:=I+1 END ;
WRITELN(U).
```

Ou, en BASIC :

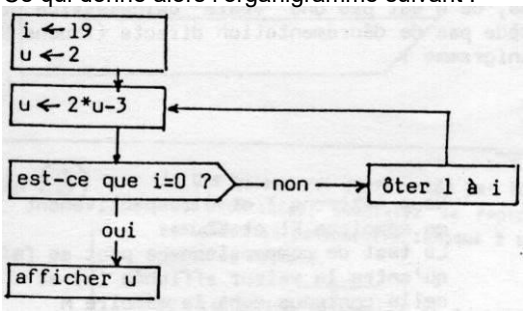
```
I=0 : U=2
WHILE I ≤ 19
U=2*U-3 : I=I+1
WEND
PRINT U
```

Cela pourrait se schématiser par l'organigramme suivant :



On "boucle" tant que i <19.
Dès que i >19 (c'est à dire i = 20) on affiche le résultat.

Au lieu de partir de i=0 et d'ajouter 1 à chaque passage, on pourrait tout aussi bien partie de i = 19 et ôter 1 jusqu'à obtenir i = 0.
Ce qui donne alors l'organigramme suivant :



Comment programmer, sur une petite machine, cette instruction conditionnelle :

SI i=0	ALORS faire :	i ← i Ŕ 1
		u ← 2u Ŕ 3
	SINON faire :	Afficher u

La plupart des calculatrices ont une touche-fonction (la touche DSZ) qui réalise directement cette instruction.
Le principe est le suivant : on met la valeur désirée (par exemple 19) dans une mémoire ; chaque fois que le programme rencontre l'instruction DSZ, il ôte 1 de te mémoire : 18, 17, etc. Quand il arrive à 0, il "saute" l'instruction suivante (qui aurait été ici le bouclage Goto) pour exécuter celle qui vient imméédiatement après (ici, l'affichage de u).

1er cas : Les CASIO 3900, 4000, 6000, 7000 etc. :

19 → I 2 → U Lbl 1 2*U Ŕ 3 → U DSZ I : Goto 1 : U◢	Les labels (Lbl) sont des étiquettes indiquant l'endroit où se rendra le programme quand il rencontrera l'instruction Goto Boucle tant que I ≠ 0 ; quand I = 0, affiche U (instruction ◢)

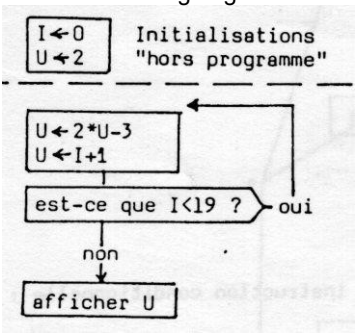
2e cas : Les TEXAS TI62, TI66 etc. :

Il faut déjà choisir le n" des mémoires dans lesquelles on stockera I et U (ici, I dans la mémoire 0 et U dans la mémoire 1).Sur la TÎ62, on n'a pas le choix de la mémoire qui sera décrémentée par DSZ : c'est obligatoirement la mémoire zéro.

19 STO0 2 STO1 Lbl1 2*RCL1 - 3 = STO1 DSZ Goto 1 RCL1 PAUSE	Boucle tant que i ≠ 0 ; quand i = 0, affiche U qui est en mémoire 1 (instruction PAUSE)

3e cas : La CASIO fx180P

Sur ce type de machine, il n'y a pas possibilité de "bouclage" sauf le retour au début de programme (en ce sens, ce n'est pas une "vraie" calculatrice programmable).
De plus, elle ne possède pas de décrémentation directe (touche DSZ). Il faut donc modifier notre organigramme :



Nous mettrons I et U respectivement en mémoires K1 et K2.
Le test de comparaison ne peut se faire qu'entre la valeur affichée (x) et celle contenue dans la mémoire M (qui devra donc contenir 19).

Le programme est alors :

2*Kout2 Ŕ 3 = Kin2 1 Kin+1 Kout1 x<M RTN Kout2	Calcule 2u-3 (dans K2) Ajoute 1 à I (dans K1) Prend la valeur de K1, la compare à M. Si c'est vrai (I<19), retour au début (RTN) sinon exécute la suite : c'est à dire ressort K2 et s'arrête.

Avant d'exécuter, initialiser par :
19 Min ; 0 Kin1 (initialise la valeur de I) ; 2 Kin2 (initialise la valeur de U)

Ces trois exemples nous montrent que, dans les calculettes "bas de gamme", une CASIO 3900P ou 400OP est plus facile à manier (car utilisant un langage beaucoup plus proche des langages évolués) qu'une TEXAS, et qu'elles l'emportent largement sur la CASIO fx180P.

REMARQUE

Les programmes écrits dans cet article ne sont pas "interactifs" : ils ne calculent que le $20^{ème}$ terme, avec $u_0 = 2$. Mais il est très facile de les modifier pour qu'ils s'arrêtent à un rang n quelconque choisi par l'utilisateur.

Sur une CASIO 400OP, par exemple, au lieu de : $19 \rightarrow I$
l'instruction aurait été : $? \rightarrow I$.
A l'exécution, la calculatrice afficherait : I ?
Il suffirait alors de taper la valeur de I (par exemple 19) et de lancer l'exécution par EXE. On pourrait faire de même avec la valeur initiale de U.