TD Info n°4 : corrigé

ECE3 Lycée Carnot

20 octobre 2011

Petits exercices

1. Cf le corrigé du TD précédent.
2. Un problème se pose pour le calcul de ce genre de suite récurrentes croisées, celui de la non-simultanéité des calculs par la machine. Si on se contente d'utiliser deux variables u et v pour représenter les termes des deux suites, il faudra à chaque étape de la boucle modifier à la fois la valeur de u et celle de v. Or, pour chacune de ces deux modifications, on a besoin des valeurs antérieures de u et de v. La première variable modifiée n'aura donc plus la bonne valeur au moment du calcul permettant de modifier la seconde. Seule possibilité : utiliser une troisième variable pour garder la valeur de la première et pouvoir la réutiliser pour le second calcul.

   ```
   PROGRAM croisees ;
   USES wincrt ;
   VAR i,n : integer ; u,v,w : real ;
   BEGIN
   WriteLn('Choisissez la valeur de n') ;
   ReadLn(n) ;
   WriteLn('Choisissez les valeurs de u0 et v0') ;
   ReadLn(u,v) ;
   FOR i :=1 TO n DO
   BEGIN
   w :=u ;
   u :=3*u/4+v/4 ;
   v :=w/4+3*v/4 ;
   END ;
   WriteLn('Les termes recherchés sont égaux à ',u,' et ',v) ;
   END.
   ```

3. Le problème est assez similaire au précédent : pour chaque nouveau calcul on a besoin des deux termes précédents de la suite. On aimerait donc utiliser deux variables u et v, qui représenteront respectivement les valeurs de u_n et de u_{n+1}. Mais alors, où stocker la valeur de u_{n+2} une fois calculée ? Le plus simple est de la mettre dans une troisième variable w, puis de décaler les valeurs (on remplace la valeur de u par celle de v et celle de v par celle de w) pour avoir la valeur de u_{n+1} dans u et celle de u_{n+2} dans v avant le calcul suivant.

   ```
   PROGRAM double ;
   USES wincrt ;
   VAR i,n,u,v,w : integer ;
   BEGIN
   WriteLn('Choisissez la valeur de n') ;
   ReadLn(n) ;
   ```

```
u :=1 ; v :=2 ;
FOR i :=1 TO n DO
BEGIN
w :=3*u-2*v ;
u :=v ;
v :=w ;
END ;
WriteLn('Le terme recherché est égal à ',u) ;
END.
```

Autre possibilité qui n'utilise que deux variables, remplacer alternativement u et v par la nouvelle valeur. Ainsi, au départ, $u = u_0$ et $v = u_1$, on remplace u pour avoir $u = u_2$ et $v = u_1$; puis on remplace v pour avoir $u = u_2$ et $v = u_3$ et ainsi de suite. Cela suppose de distinguer des cas suivant que i est pair ou non, ce qui se teste via la commande $odd(i)$ (qui est vraie si i est impair, fausse sinon).

```
PROGRAM double2 ;
USES wincrt ;
VAR i,n,u,v : integer ;
BEGIN
WriteLn('Choisissez la valeur de n') ;
ReadLn(n) ;
u :=1 ; v :=2 ;
FOR i :=1 TO n DO
IF odd(i) THEN u :=3*u-2*v ELSE v :=3*u-2*v ;
IF odd(n) THEN WriteLn(u) ELSE WriteLn(v) ;
END.
```

4. Ce programme tire un nombre au hasard entre 0 et 10 et laisse cinq essais au joueur pour le deviner.
5.
```
PROGRAM jeu2 ;
USES wincrt ;
VAR i,n,p : integer ;
BEGIN
Randomize ;
n := random(101) ;
FOR i :=1 TO 5 DO
BEGIN
WriteLn('Choisissez un nombre entre 0 et 100') ;
ReadLn(p) ;
IF p=n THEN
BEGIN
WriteLn('Bravo, vous avez gagné') ;
i :=5 ;
END
ELSE
BEGIN
WriteLn('Pas de chance !') ;
IF p<n THEN WriteLn('Nombre tenté trop petit') ELSE WriteLn('Nombre tenté trop grand') ;
```

```
    END ;
    END ;
    WriteLn('Le jeu est terminé') ;
    END.
```

6. Le mieux est de procéder par dichotomie, c'est-à-dire à chaque étape de tenter le nombre le plus proche du centre de l'intervalle où l'on sait que se situe le nombre à trouver. Ainsi, au premier essai, on sait que n est compris entre 0 et 100, on vise au milieu, c'est-à-dire 50. Imaginons que le nombre tenté soit trop petit. On sait désormais que n est entre 50 et 100, on vise alors au milieu, c'est-à-dire 75, et ainsi de suite. Je vous laisse vérifier si vous le souhaitez qu'on mettra au plus 7 essais à trouver le nombre ainsi. Par exemple, si ce nombre est 76, on tentera 50, 75, 88, 82, 79, 77 et enfin 76 (à certains moments, on a le choix entre deux nombres adjacents car le centre de l'intervalle n'est pas un nombre entier).

   Si on monte jusqu'à un million de possibilités, une vingtaine d'essais au maximum seront suffisants. En effet, après le premier essai, on est à une distance maximale du bon nombre égale à 500 000. Après le deuxième essai, cette distance est divisée par 2, on est à moins de 250 000 du nombre à deviner, et ainsi de suite (en oubliant les arrondis dus aux essais ne tombant pas juste à la moitié. Au bout de 20 essais, on est à une distance inférieure à $\frac{1\ 000\ 000}{2^{20}} \simeq 0.95$ du résultat, c'est-à-dire qu'on est nécessairement sur le bon résultat puisqu'on ne travaille qu'avec des nombres entiers. Plus généralement, si on a n valeurs possibles au départ, le nombre maximal d'essais nécessaires en utlisant la dichotomie sera environ $\log_2(n)$.