TD ESSTIN 4 (exercices sur les boucles)

Vincent Thomas (thomasv0@cti.ecp.fr)

23 octobre 2000

1 Exercice pour se chauffer les meninges

1.1 Premier exercice sur les boucles

Ecrire un programme qui affiche les entiers de 1 à n en utilisant :
- une boucle **pour**
- une boucle **tant que**

1.2 Addition et multiplication

Ecrire un algorithme qui calcule la multiplication du réel **a** par un entier **n** en n'utilisant que l'addition.

1.3 Un exercice classique

Ecrire un algorithme qui calcule factorielle de **n**

1.4 Etude des algorithmes suivants

Que font les deux algorithmes suivants ?

```
VARIABLES
    Réel a
    Entier n
    Entier parcours
    Réel resultat
DEBUT
    Lire (a)
    Lire (n)
    resultat ← a
        pour parcours allant de 1 à n
            resultat ← resultat.a
        fin pour
    Affiche (résultat)
FIN
```

```
VARIABLES
    Réel a
    Entier n
    Entier parcours
    Réel resultat
    Réel temp
DEBUT
    Lire (a)
    Lire (n)
    temp ← 1
    resultat ← a
        tant que (n ≠ 1)
            si (npair)
                alors
                    resultat ← resultat.resultat
                    n ← n/2
                Sinon
                    temp ← temp.resultat
                    n ← n − 1
            fin si
        fin tant que
    resultat ← resutat.temp
    Affiche (resultat)
FIN
```

Quel est l'interet du second algorithme par rapport au premier ?

2 De la suite dans les idéees

Considérons U_n une suite finie (c'est à dire nulle à partir d'un certain rang). Cette suite sera rangée dans un tableau de longueur l (avec l bien entendu superieur à la longueur de la suite). Le but va consister à parcourir la liste afin de déterminer certaines de ses caracteristiques.

Pour les questions suivantes, l'exemple utilisé sera la suite $U_n = \{4, 2, 2, 3, 3, 3, 2, 1, 4, 5, 3, 3, 5\}$

2.1 Lecture

Ecrire l'algorithme qui permet de demander une suite à l'utilisateur de deux manières :
- En demandant le nombre de valeurs
- En considérant qu'un 0 implique la fin de la suite

Ecrire une fonction qui permette d'afficher la suite.

2.2 Moyenne

Ecrire un algorithme qui permette de calculer la moyenne de la suite.
L'algorithme donnera pour réponse 3.0769 à notre exemple.

2.3 Minimum et maximum

Ecrire un algorithme qui renvoie le maximum et le minimum de la liste.
Pour notre exemple, l'algorithme doit renvoyer : $maximum = 5, minimum = 1$

2.4 Différentes valeurs

On suppose dés lors que la liste est constituée d'entiers compris entre 1 et n (n supposé connu).

Ecrire un algorithme qui renvoie les différentes valeurs de la liste.

Pour notre exemple, n vaut 5 (ou plus), et l'algorithme donne comme réponse : $\{1, 2, 3, 4, 5\}$ (note : on ne se préoccupe pas de l'ordre des éléments).

2.5 Nombre d'occurences

Ecrire un algorithme qui affiche le nombre d'occurences de chaque élément.
Pour notre exemple, la réponse sera : $\{1(1), 2(3), 3(5), 4(2), 5(2)\}$

2.6 Sous suite

Ecrire un algorithme qui extrait la plus longue sous-suite composée d'éléments constants consécutifs ainsi que le nombre d'éléments et la position de cette sous-suite dans la suite.

Pour notre exemple, la réponse que l'algorithme devra fournir sera : la suite composée de 3, de longueur 3 et de début 4 dans la suite U_n.

2.7 Question Subsidiaire

Quel est l'interet de limiter le nombre de valeurs possibles de la suite ?

3 Prérequis pour la suite

Avant d'aller plus loin, résoudre les deux problemes suivants qui pourront être utilisés dans les exercices du prochain chapitre.

3.1 Reflexions sur l'affichage

On souhaite avoir un affichage plus compréhensible et plus visuel que ce qui a été fait jusqu'à présent. Pour cela, on va se donner un petit problème à résoudre , l'important étant la manière dont l'affichage se fera (avec l'appel à la fonction Afficher).

Le but est d'écrire un algorithme qui permette de simuler le déplacement d'une particule (ou d'une bestiole, c'est vous qui voyez...(appelons la Bebert)) le long d'un axe. Bebert donc, peut se déplacer vers la gauche ou vers la droite d'un nombre de cases quelconques. Simplement, l'environnement dans lequel il évolue est fini et constitué de n cases (n étant donné) alignées.

Il va falloir écrire l'algorithme qui simule le déplacement de Bebert en fonction de l'ordre qu'on lui donne. On affichera, aprés chaque ordre, les cases de l'environnement à l'aide de

caractères. le caractère 'o' désignera une case vide et le caractère 'x' la case ou Bebert se trouve.

Par exemple : (environnement composé de 10 cases)

$$oooooxooo \rightarrow^{g2} ooooxooooo \rightarrow^{d4} ooooooooxo \rightarrow^{d3} ooooooooox$$

3.2 Tableaux à deux dimensions (voire plus)

Les tableaux sont définis à partir d'un type quelconque. Ainsi on peut définir des tableaux d'entiers, de réels, de caractères voire meme de tableaux. Et ainsi de suite ... ce qui permet de définir des tableaux de taille et de nombre de dimensions quelconques. (note : Quelques utilisations des tableaux à plusieurs dimensions, tableaux à deux dimensions : matrice, image, points dans le plan, tableaux à 3 dimensions : points dans l'espace,..)

Par exemple int Tableau[2][5] est un tableau à deux dimensions. On accède à l'element souhaité par Tableau[i][j] (avec $i \in [0, 1]$ et $j \in [0, 4]$).

Afin de se familiariser avec de tels tableaux écrivez une fonction qui affiche le contenu d'un tableau à deux dimensions sous la forme d'une matrice.

4 Le jeu de la vie

4.1 Principe

Les règles du jeu de la vie (inventé par John H.Conway) dont vous allez écrire l'algorithme sont assez simples. Ce "jeu" simule l'évolution de cellules dans un environnement composé de cases. Il s'agit de donner une configuration initiale et de laisser évoluer le système.

L'environnement est un environnement discrétisé à deux dimensions (pour simplifier, il s'agit d'un tableau à deux dimensions). Chaque case du tableau peut etre vide ou occupée par une cellule (et une seule !). Les cellules évoluent en fonction des 8 voisins qui l'entourent.

Pour culture, le jeu de la vie touche à de nombreux domaines

- La logique et la théorie du chaos : Bien que les cellules possèdent des comportements simples, l'évolution du systeme est pratiquement indécidable.
- La biologie : Des cellules naissent et meurent, le système correspond à une simulation d'écosystème : un ensemble d'individus évolue simultanément avec ses problèmes de ressources, de proies, de prédateurs, etc...
- Les mathématiques : Convergence ou divergence du système ?

4.2 Les règles à appliquer

Les règles qui regissent l'évolution des cellules entre deux étapes sont les suivantes :

- **Règle 1, règle de la survie :** Chaque cellule, ayant à l'étape n, deux ou trois cellules adjacentes survit à l'étape n+1
- **Règle 2, règle de la mort :** Chaque cellule ayant pour voisines, à l'étape n, quatre cellules ou plus meurt par etouffement (surpopulation) à l'étape n+1. De meme, une cellule qui n'a qu'une ou aucune cellule adjacente meurt par isolement.

– **Règle 3, règle des naissances :** Chaque case vide, ayant exactement trois cellules adjacentes à l'étape n engendre une nouvelle cellule à l'étape n+1.

4.3 Votre travail

Ecrire l'algorithme qui simule le jeu de la vie (pour simplifier, on peut tout d'abord écrire l'algorithme consistant à determiner l'évolution d'une cellule et ensuite gérer le système dans sa globalité).

Ecrire l'algorithme qui se charge de demander à l'utilisateur une configuration initiale et d'afficher les résultats (cf partie 3). L'affichage consistera à un carré englobant l'environnement et, pour l'environnement, à afficher un espace (' ') pour les cases vides et un 'O' pour les cellules.

4.4 exemple

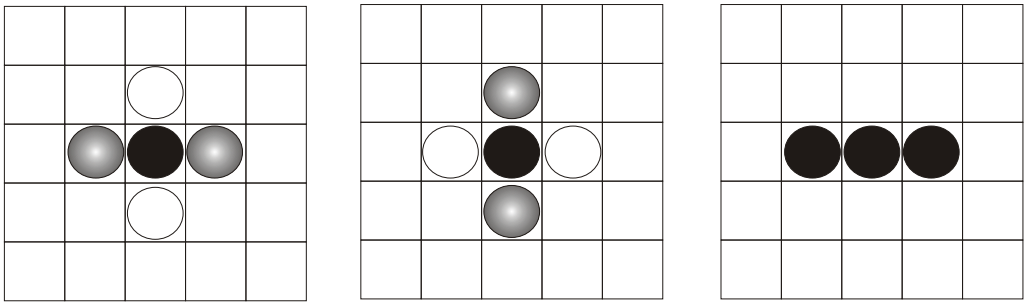

FIG. 1 – Le "clignotant"

Dans cet exemple :
– les ronds noirs représentent des cellules de l'étape n, encore présentes à l'étape n+1
– les ronds blancs représentent des générations spontanées
– les ronds noir-blanc représentent une cellule qui va mourir