4 – Introduction aux algorithmes itératifs

Notations

Notion de séquence

- En informatique, on peut être amené à répéter des actions plusieurs fois.
- On raisonne donc sur une séquence, c'est-à-dire une suite d'éléments de même type, et de valeurs différentes

  ○ ○ ○ ○ … ○ ○ ○

- Exemples de séquences : chaines de caractères, fichiers, tableaux, suite mathématique, etc.
- Ces séquences peuvent être représentées de diverses manières sur le plan informatique.
- Un algorithme itératif consiste à énumérer les éléments d'une séquence.

Enumération des éléments d'une séquence

- Il y a en fait deux grandes raisons pour énumérer les éléments d'une séquence :
  1. Appliquer **un même traitement à tous les éléments** de la séquence = **parcours séquentiel**
  2. Rechercher dans la séquence un élément vérifiant une propriété donnée = **recherche séquentielle**
- Tous les algorithmes itératifs consistent
  - soit en un parcours d'une séquence
  - soit en une recherche dans une séquence
  - soit en une combinaison de parcours et de recherches

Formes de composition itérative

Répéter n fois

```
A
répéter n fois  // n est une expression entière ≥ 0
    B    //composition d'actions
frépéter
C
```

On effectue l'action A puis n fois l'action B, puis C

Formes de composition itérative

Tant que

```
A
tantque α faire // α est une expression  logique
    B    //composition d'actions
ftantque
C
```

- α est appelée **condition de continuation** de l'itération
- On effectue l'action A, puis zéro, une ou plusieurs fois l'action B, puis C.
- Si α a la valeur faux dès le départ, on n'effectue jamais B
- L'action B est effectuée jusqu'à ce que α ait la valeur faux
- Que se passe-t-il si l'action B ne modifie pas la condition α ?

Formes de composition itérative

Répéter

```
A
répéter
    B    //composition d'actions
jusqu'à β
C
```

- β est appelée **condition d'arrêt** de l'itération
- On effectue l'action A, puis une ou plusieurs fois l'action B, puis C
- L'action B est effectuée au-moins une fois et jusqu'à ce que β ait la valeur vrai
- Si β a la valeur vrai dès le départ, on effectue une seule fois B
- Si au départ β a la valeur faux, et si l'action B ne modifie pas la condition β on a une itération sans fin : on dit que l'algorithme « boucle ».

Formes de composition itérative

Pour

```
A
pour i allant de ideb à ifin
    B    // composition d'actions
fpour
C
```

- i est appelé **indice de parcours** de l'itération, c'est un entier défini entre **ideb** et **ifin+1.**
- On effectue l'action A, puis ifin-ideb+1 fois l'action B, puis C
- La première fois que B est exécutée, i vaut ideb, la seconde fois ideb+1, etc., la dernière fois i vaut ifin.
- Si ifin-ideb+1 ≤ 0 l'action B n'est jamais exécutée.

Formes de composition itérative

Equivalent du pour

```
A
i ← ideb
tantque i ≤ ifin faire
    B    //composition d'actions
    i ← i + 1
ftantque
// i = ifin+1
C
```

Formes de composition itérative

Pour généralisé :

```
A
pour i allant de ideb à ifin pas p
    B    // composition d'actions
fpour
C
```

- l'indice i de parcours de l'itération est incrémenté de la valeur p à chaque itération
- La première fois que B est exécutée, i vaut ideb, la seconde fois ideb+p, la troisième fois ideb+p+p, etc.
- l'action B n'est jamais exécutée si l'indice de fin est atteint dès le départ :
  - si p > 0 et ideb > ifin
  - si p < 0 et ideb < ifin

Formes de composition itérative

Equivalent du pour généralisé

si pas > 0

```
A
i ← ideb
tantque i ≤ ifin faire
      B
      i ← i + pas
ftantque
// i > ifin
C
```

si pas < 0

```
A
i ← ideb
tantque i ≥ ifin faire
      B
      i ← i + pas
ftantque
// i < ifin
C
```

Exemple 1 : algorithme qui affiche les n premiers entiers naturels

```
lexique principal
 cl : clavier  // périphérique d'entrée
 e : écran     // périphérique de sortie
 n : entier ≥ 0   // donnée : nombre d'entiers à afficher
 k : entier ≥ 0   // intermédiaire : nombre d'entiers affichés
Algorithme principal
 cl.saisir(n)
 k ← 0
 tantque k ≠ n faire    // on a affiché k nombres
    // k = k0 et on a affiché les k0 premiers entiers
    e.afficher(k)
    k ← k+1
    // k = k0+1 et on a affiché les k0+1 premiers entiers
 fintantque
 // k = n  => on a affiche les n premiers entiers
 e.afficher("terminé !")
```

Exemple 2 : fonction qui calcule le nombre de ‘a’ présents dans une chaîne

```
Fonction nba(x : chaine) → entier ≥ 0
// nba(x) renvoie le nombre de ‘a’ présents dans la chaine x
// x : paramètre : chaine examinée
Lexique de nba
 na : entier ≥ 0   // valeur renvoyée : nombre de ‘a’ rencontrés dans x
 i : entier ≥ 0    // intermédiaire : indice de parcours de x
Algorithme de nba
 i ← 0
 na ← 0
 tantque i < longueur(x) faire
    si nième(x,i) = ‘a’ ou nième(x,i) = ‘A’
    alors
          na ← na +1
    fsi
    i ← i +1
 fintantque
 renvoyer(na)
```