Mise en œuvre des schémas

Jean-Michel Adam - UGA - UFR SHS

Exemple 1

- Ecrire un algorithme qui affiche un message indiquant si un texte représenté dans un fichier comporte au-moins **n** occurrences du caractère **c**.
- n et c sont des données du problème

Recherche ou parcours ?

Parcours d'une sous-séquence

- Une sous-séquence est une partie de séquence, sa fin est caractérisée
  - Soit par la fin de la séquence
  - Soit par une propriété particulière indiquant la fin de la sous-séquence
- Dans notre exemple on va parcourir la séquence de caractères jusqu'à ce que
  - soit on ait atteint la fin de fichier
  - soit on ait rencontré n occurrences de c

Au-moins **n** occurrences de **c**

```
lexique principal
f : fichier de caractères  // donnée : texte examiné
cl : clavier               // périphérique de saisie
e : écran                  // périphérique de sortie
c : caractère              // donnée : caractère recherché
n : entier > 0             // donnée : nombre d'occurrences de c recherchées
nc : entier ≥ 0            // inter : nombre d'occurrences de c de la partie parcourue de f
algorithme principal
  cl.saisir(c,n)
  f.lirePremier
  nc ← 0
  tantque non f.fdf et nc ≠ n faire
        si f.ec = c alors nc ← nc + 1 fsi
        f.lireSuivant
  ftantque
  // f.fdf ou nc = n
  selon n, nc
     nc = n : e.afficher("au-moins " , n, " occurrences de ", c)
     nc ≠ n : e.afficher("seulement  ", nc, " occurrences de ", c)
  fselon
```

Exemple 2

- Ecrire un algorithme qui calcule la longueur du premier mot d'un texte représenté dans un fichier de caractères.

Longueur du premier mot d'un texte : recherche puis parcours d'une sous-séquence

```
lexique principal
   f : fichier de caractères  // donnée : texte examiné
   e : écran                  // périphérique de sortie
   lpm : entier ≥ 0           // inter: longueur de la partie parcourue du premier mot
algorithme principal
  // étape 1 : ignorer les espaces éventuels qui précèdent le premier mot
  f.lirePremier
  tantque non f.fdf etpuis f.ec = ' ' faire  // recherche du 1er caractère du 1er mot
        f.lireSuivant
  ftantque
  // f.fdf oualors f.ec ≠ ' '
  selon f
     f.fdf : e.afficher("aucun mot dans le fichier ")
  ¬ f.fdf :      // f.ec est le 1er caractère du premier mot
                 lpm ← 0
                 répéter
                    lpm ← lpm + 1
                    f.lireSuivant
                 jusqu'à f.fdf oualors f.ec = ' '
                 // f.fdf oualors f.ec = ' '
                e.afficher("longueur du premier mot :", lpm)
  fselon
```

Exemple 3

- Ecrire un algorithme qui recopie une séquence d'entiers représentée dans un fichier, dans un tableau d'entiers comportant LMAX éléments.
- On affichera un message avec la longueur de la séquence mémorisée, et le cas échéant, on indiquera si la séquence a dû être tronquée.

Copie d'une séquence fichier dans un tableau : parcours d'une sous-séquence

```
Lexique principal
  f : fichier d'entiers  // séquence à mémoriser dans t
  t : tableau sur [0...LMAX-1] d'entiers
  lg : entier entre 0 et LMAX       // longueur de la séquence mémorisée
  k : entier entre 0 et LMAX        // indice du 1er emplacement libre dans t
Algorithme principal
  f.lirePremier
  k ← 0
  tantque non f.fdf et k < LMAX faire
     t[k] ← f.ec
     k ← k + 1
     f.lireSuivant
  ftantque
  // f.fdf ou k = LMAX
  lg ← k
  si non f.fdf
  alors  e.afficher("séquence mémorisée tronquée à ",LMAX," entiers" )
  sinon e.afficher("séquence mémorisée de longueur ",lg)
  fsi
```

Diverses représentation de séquences

- **Fichier séquentiel** : séquence fournie sur un support externe
- **Séquence représentée dans un tableau**: deux sens de parcours possibles
- **Chaine de caractères** : deux sens de parcours possibles
- **Séquences calculées** : énumération d'une suite de valeurs calculées, jusqu'à obtenir une valeur vérifiant une propriété et caractérisant la fin de séquence.

Fichier séquentiel

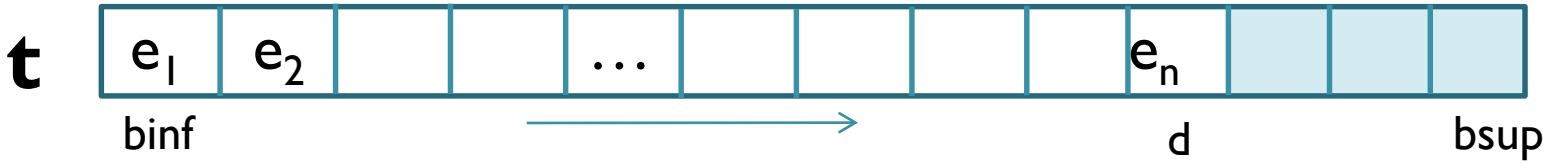
Lexique

f : fichier de typedeséléments

EC	f.ec
FDS	f.fdf
AV	f.lireSuivant
DEM	f.lirePremier

Séquence représentée dans un tableau

énumération de binf vers bsup



Lexique

t : tableau sur [binf..bsup] de typedeséléments

i : entier entre binf et bsup+1 // indice de parcours de t

d : entier entre binf-1 et bsup // position du dernier élément de t (longueur de la séquence -1)

EC	t[i]
FDS	i > d
AV	i ← i + 1
DEM	i ← binf

Séquence représentée dans un tableau

énumération en sens inverse : de bsup vers binf

Lexique

t : tableau sur [binf..bsup] de typedeselements

i : entier entre binf-1 et bsup // indice de parcours de t

d : entier entre binf-1 et bsup // position du dernier élément de t (longueur de la séquence -1)

EC	t[i]
FDS	i < binf
AV	i ← i - 1
DEM	i ← d

La séquence est vide si d = binf-1

Au-moins **n** occurrences de **c**

séquence représentée dans un tableau

```
lexique principal
t : tableau sur [0..LMAX-1] de caractères  // texte examiné
i : entier entre 0 et LMAX   // indice de parcours de t
d : entier entre -1 et LMAX-1      // position du dernier élément de t (longueur du texte-1)
cl : clavier            // périphérique de saisie
e : écran                     // périphérique de sortie
c : caractère               // donnée : caractère recherché
n : entier > 0              // donnée : nombre d'occurrences de c recherchées
nc : entier ≥ 0             // inter : nombre d'occurrences de c rencontrées

Algorithme principal
  cl.saisir(c,n)
  i ← 0
  nc ← 0
  tantque i ≤ d et nc ≠ n faire
        si t[i] = c alors nc ← nc + 1 fsi
        i ← i + 1
  ftantque
  // i > d ou nc = n
  selon n, nc
        nc = n :    e.afficher("au-moins" , n, " occurrences de ", c)
        nc ≠ n :    e.afficher("seulement  ", nc, " occurrences de ", c)
  fselon
```

Chaine de caractères

énumération du début à la fin

Lexique

s : chaine de caractères // séquence à examiner

i : entier ≥ 0 // indice de parcours de s

EC	nième(s,i)
FDS	i = longueur(s)
AV	i ← i + 1
DEM	i ← 0

Chaine de caractères

énumération en sens inverse : de la fin vers le début

Lexique

s : chaine de caractères // séquence à examiner

i : entier ≥ -1 // indice de parcours de s

EC	nième(s,i)
FDS	i < 0
AV	i ← i - 1
DEM	i ← longueur(s) - 1

Au-moins **n** occurrences de **c**

séquence représentée dans une chaine

```
lexique principal
texte : chaine        // texte examiné
i : entier ≥ 0              // indice de parcours de texte
cl : clavier                // périphérique de saisie
e : écran             // périphérique de sortie
c : caractère               // donnée : caractère recherché
n : entier > 0              // donnée : nombre d'occurrences de c recherchées
nc : entier ≥ 0             // inter : nombre d'occurrences de c rencontrées

Algorithme principal
  cl.saisir(c,n) ; cl.saisir(texte)
  i ← 0
  nc ← 0
  tantque i < longueur(texte) et nc ≠ n faire
      si nième(texte,i) = c alors nc ← nc + 1 fsi
       i ← i + 1
  ftantque
  // i = longueur(texte) ou nc = n
  selon n, nc
       nc = n :    e.afficher("au-moins" , n, " occurrences de ", c)
       nc ≠ n :    e.afficher("seulement  ", nc, " occurrences de ", c)
  fselon
```

Séquence calculée

énumération d'une suite de valeurs calculées, jusqu'à obtenir une valeur de fin vérifiant une propriété P

Lexique

v : type des éléments // élément courant de la suite

EC	v
FDS	P(v)
AV	$v \leftarrow f(v)$
DEM	$v \leftarrow v_{init}$

Séquence calculée : exemple
énumération des multiples de 3 inférieurs à 100

Lexique

m : entier ≥ 0 // multiple de 3 courant

EC	m
FDS	m ≥ 100
AV	m ← m + 3
DEM	m ← 0

Séquence calculée : exemple

énumération des puissances de 2 inférieures à 1000

Lexique

p : entier ≥ 0 // puissance de 2 courante

EC	p
FDS	p ≥ 1000
AV	p ← p * 2
DEM	p ← 1

// p = $2^k * 2 = 2^{k+1}$

// p = 2^0