Université Bordeaux 1 — Licence Informatique 2003/2004

Initiation à l'algorithmique

Corrigé Contrôle Continu du 2 avril 2004
Sujet 1

Exercice .1

```
fonction  Mystère(ref T: tableau[0..NMAX-1] de booléens,
                  val N: entier): entier;
  var i, k, m: entier;
  début
       i = 0;
       m = 0;
       tantque i<N faire
         k = 0;
         tantque i<N  ET  T[i]==vrai faire
           k = k+1;
           i = i+1;
         fintantque;
         si k>m alors
           m = k;
         finsi;
         i = i+1;
       fintantque;
       retourne(m);
  fin
finFonction
```

1. On considère le tableau `T` contenant la suite

$$vrai, faux, faux, vrai, vrai, vrai, faux, vrai, vrai$$

et la variable $n = 9$.

Quel est l'effet de l'appel `Mystère(T,n)` ? Justifiez votre réponse.

Trace de l'exécution :

```
i=0  m=0
          i < N ?? oui
             k=0
             i < N  ??  oui  T[0] == vrai ?? oui
               k=1
               i=1
             i < N  ??  oui  T[1] == vrai ?? non
             k > m ?? oui
               m=1
i=2
          i < N ?? oui
             k=0
             i < N  ??  oui  T[2] == vrai ?? non
             k > m ?? non
```

```
i=3
         i < N ?? oui
            k=0
            i < N  ??  oui  T[3] == vrai ?? oui
              k=1
              i=4
            i < N  ??  oui  T[4] == vrai ?? oui
              k=2
              i=5
            i < N  ??  oui  T[5] == vrai ?? oui
              k=3
              i=6
            i < N  ??  oui  T[6] == vrai ?? non
            k > m ?? oui
              m=3
i=7
         i < N ?? oui
            k=0
            i < N  ??  oui  T[7] == vrai ?? oui
              k=1
              i=8
            i < N  ??  oui  T[8] == vrai ?? oui
              k=2
              i=9
            i < N  ??  non  (deuxième test non evalué)
            k > m ?? non
i=10
         i < N ?? non
         retourne (3)
```

L'appel de `Mystère(T,n)` a comme effet de retourner la valeur 3 : celle-ci est la longueur de la plus longue séquence de valeurs *vrai* contenue dans le tableau `T` donné en exemple.

2. En général, que fait la fonction `Mystère` ?

   La fonction `Mystère` calcule (et retourne) la longueur de la plus longue suite de valeurs *vrai* dans le tableau. La variable `k` va successivement contenir les longueurs des suites de *vrai*, la variable `m` n'est modifiée que lorsqu'on trouve une suite de *vrai* de longueur supérieure aux précédentes.

3. Quel est le contenu du tableau `T` si la fonction `Mystère(T,n)` retourne la valeur 0 ? Et si elle retourne n ?

   Si elle retourne 0 le tableau ne contient que des valeurs $faux$, si elle retourne n le tableau ne contient que des valeurs $vrai$.

4. Quel est le nombre maximal d'affectations effectuées par cet algorithme ? (Justifiez brièvement la réponse).

   On va donner une borne supérieure au nombre d'affectations. Hormis les deux affectations initiales, on s'intéresse aux nombres d'affectations selon la valeur de `T[i]` sachant la valeur de `T[i-1]`. Il y a quatre cas différents :
   1 `T[i-1]=F T[i]=F` $\longrightarrow$ 2 affectations (`k=0` et `i=i+1`)
   2 `T [i-1]=V T[i]=F` $\longrightarrow$ 1 ou 2 affectations (`i=i+1` et `m=k` si nécessaire)
   3 `T [i-1]=F T[i]=V` $\longrightarrow$ 3 affectations (`k=0`, `k=k+1`, `i=i+1`)

4 `T [i-1]=V T[i]=V` $\longrightarrow$ 2 affectations (`k=k+1`, `i=i+1`)
Pour `i=0` on est :
- soit dans le cas 1(`T[0]=F`)
- soit dans le cas 3 (`T[0]=V`).

Le cas 1 indique que l'on est en dehors d'une série de 1
Le cas 2 indique que l'on termine une série de 1
Le cas 3 indique que l'on commence une série de 1
Le cas 4 indique que l'on est à l'intérieur d'une série de 1

Chaque transition (`T[i-1]` à `T[i]`) coute au plus 3 affectations. Il y a N transitions donc au maximum le nombre d'affectations est inférieur à $3N + 2$.

5. Indiquer l'ordre de complexité de l'algorithme.
   Il s'agit d'un parcours de tableau : chaque élément du tableau est testé une et une seule fois. La compléxité est donc $O(n)$.

Exercice .2 Quelle est, parmi les trois méthodes élémentaires de tri (séléction, insertion, bulle), celle qui a la même complexité dans le meilleur des cas et dans le pire des cas ? Justifiez brièvement votre réponse.
Le tri sélection a la même complexité dans le meilleur des cas (suite déjà triée) et dans le pire des cas (suite dans l'ordre inverse) : il fait toujours $n * ((n - 1)/2)$ comparaisons (dans la fonction qui recherche le minimum) et n échanges.

Exercice .3

Faire tourner l'algorithme du **`tri à bulle`** sur le tableau contenant la suite $9, 10, 1, 5, 12, 7, 3, 8$.
Décrire l'évolution du contenu du tableau et des variables de l'algorithme lors de l'exécution de la fonction.
On considère que le tableau contenant la suite est indexé de 1 à 8.

Trace d'exécution :
`i=1`
`pour j allant de 8 à 2 par pas de -1` :

$$9, 10, 1, 5, 12, 7, 3, 8$$
$$9, 10, 1, 5, 12, 3, 7, 8$$
$$9, 10, 1, 5, 3, 12, 7, 8$$
$$9, 10, 1, 3, 5, 12, 7, 8$$
$$9, 10, 1, 3, 5, 12, 7, 8$$
$$9, 1, 10, 3, 5, 12, 7, 8$$
$$1, 9, 10, 3, 5, 12, 7, 8$$

`i=2`
`pour j allant de 8 à 3 par pas de -1` :

$$1, 9, 10, 3, 5, 12, 7, 8$$
$$1, 9, 10, 3, 5, 7, 12, 8$$

$$1, 9, 10, 3, 5, 7, 12, 8$$

$$1, 9, 10, 3, 5, 7, 12, 8$$

$$1, 9, 3, 10, 5, 7, 12, 8$$

$$1, 3, 9, 10, 5, 7, 12, 8$$

`i=3`
`pour j allant de 8 à 4 par pas de -1` :

$$1, 3, 9, 10, 5, 7, 8, 12$$

$$1, 3, 9, 10, 5, 7, 8, 12$$

$$1, 3, 9, 10, 5, 7, 8, 12$$

$$1, 3, 9, 5, 10, 7, 8, 12$$

$$1, 3, 5, 9, 10, 7, 8, 12$$

`i=4`
`pour j allant de 8 à 5 par pas de -1` :

$$1, 3, 5, 9, 10, 7, 8, 12$$

$$1, 3, 5, 9, 10, 7, 8, 12$$

$$1, 3, 5, 9, 7, 10, 8, 12$$

$$1, 3, 5, 7, 9, 10, 8, 12$$

`i=5`
`pour j allant de 8 à 6 par pas de -1` :

$$1, 3, 5, 7, 9, 10, 8, 12$$

$$1, 3, 5, 7, 9, 8, 10, 12$$

$$1, 3, 5, 7, 8, 9, 10, 12$$

`i=6`
`pour j allant de 8 à 7 par pas de -1` :
pas d'échanges : la suite est triée

Exercice .4

1. Ecrire une fonction calculant le nombre d'éléments minimaux et l'indice de la première occurrence d'un élément minimal dans un tableau contenant une suite de N nombres entiers.

```
fonction nbMiniEtIndicePremier(ref T:tableau[0..NMAX-1] d'entiers,
                               ref compteur, indice_min:entier,
                               val N:entier):vide;
  var i:entier;
  début
    compteur = 1;
    indice_min = 0;
    pour i allant de 1 à N-1 faire
      si T[i]<T[indice_min] alors
        indice_min = i;
        compteur = 1;
      sinon
        si T[i]==T[indice_min] alors
          compteur = compteur + 1;
      finsi;
    finpour;
  fin
finFonction
```

2. Ecrire une fonction calculant les deux plus petits éléments distincts dans un tableau de N entiers.

```
fonction deuxPlusPetits(ref T:tableau[0..NMAX-1] d'entiers,
                        ref min1,min2:entier,
                        val N:entier):vide;
  var i:entier;
  début
    i=1;
    tantque T[i]==T[0] et i<N-1 faire
      i=i+1;
    fintantque;
    si T[i]==T[0] alors
      min1=min2=T[0]; // tous les éléments sont égaux//
    sinon
      si T[0]<T[i] alors
        min1=T[0];
        min2=T[i];
      sinon
        min1=T[i];
        min2=T[0];
      finsi;
      i=i+1;
      tantque i<N faire
        si T[i]<min1 alors
          min2=min1;
          min1=T[i];
        sinon
           si T[i]<min2 et T[i]<>min1 alors
             min2=T[i];
           finsi;
        finsi;
        i=i+1;
      fintantque;
   finsi;
  fin
finFonction
```