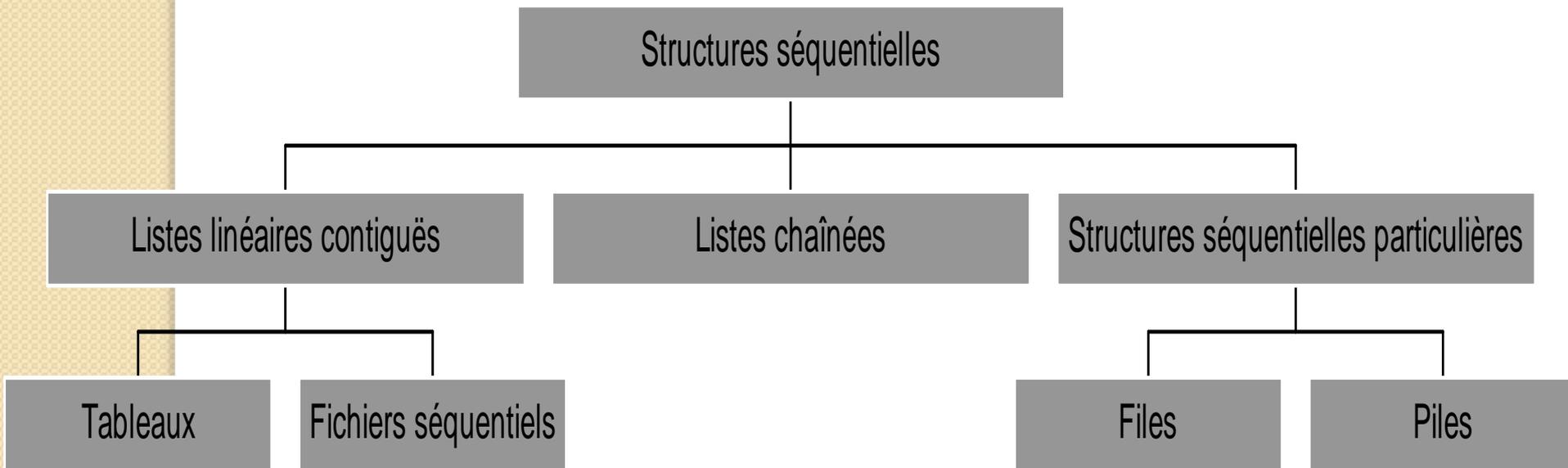


Les structures de données séquentielles contiguës

Isabelle Comyn-Wattiau

- 1. Les structures séquentielles
- 2. Les listes linéaires contiguës
- 3. Les tableaux
- 4. Les fichiers séquentiels

I. Les structures séquentielles



Notion de structure

- Rôle des variables dans les programmes
- Une structure = un ensemble de variables de même nature auxquelles on doit accéder :
 - soit directement par leur numéro d'ordre
 - soit en les parcourant une par une dans l'ordre

Parcours d'une structure

- Recherche séquentielle : examiner tous les éléments jusqu'à trouver celui que l'on cherche
- Accès direct : on peut directement examiner l'élément que l'on souhaite - impossible dans une structure purement séquentielle
- Recherche de type dichotomique : possible dans une structure séquentielle triée

Types de structures séquentielles

- Liste linéaire : suite d'éléments de même type
- Exemples : fichier séquentiel, tableau, file, pile
- 2 organisations
 - contiguë
 - chaînée

Parcours séquentiel d'une liste linéaire L

- Supposons que l'on ait :
 - une fonction tête pour accéder au premier élément
 - une fonction longueur pour connaître le nb d'éléments
 - une fonction succ pour accéder à l'élément suivant

- Complexité : $o(n)$

- sauf si...

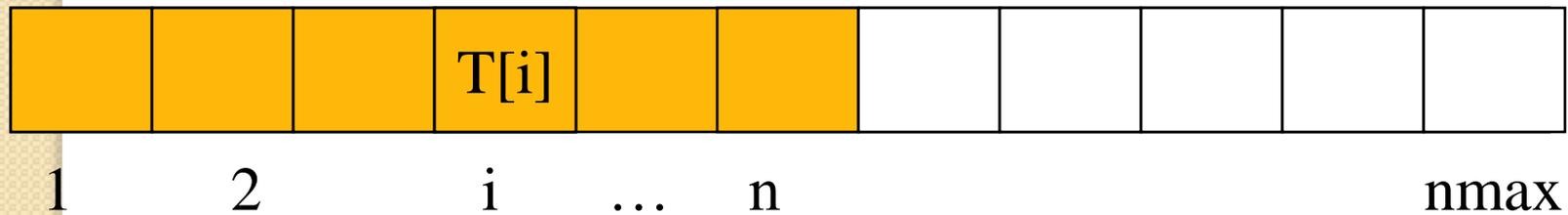
```
X:=tête(L);  
traiter(X);  
pour i:=1 jusqu'à longueur(L)-1 faire  
    X:=succ(X);  
    traiter(X);  
finpour
```



2. Les listes linéaires contiguës

- Suite d'éléments de même nature
- stockés les uns à la suite des autres
- l'accès est au minimum séquentiel
- mais d'autres accès sont parfois possibles

3. Le vecteur ou tableau



- **Parcours d'un tableau**

$i:=1;$

Tantque $i \leq n$ faire

 traiter $T[i];$

$i:=i+1;$

fintantque

- **Somme des éléments**

$S:=0; i:=1;$

Tantque $i \leq n$ faire

$S:=S+T[i];$

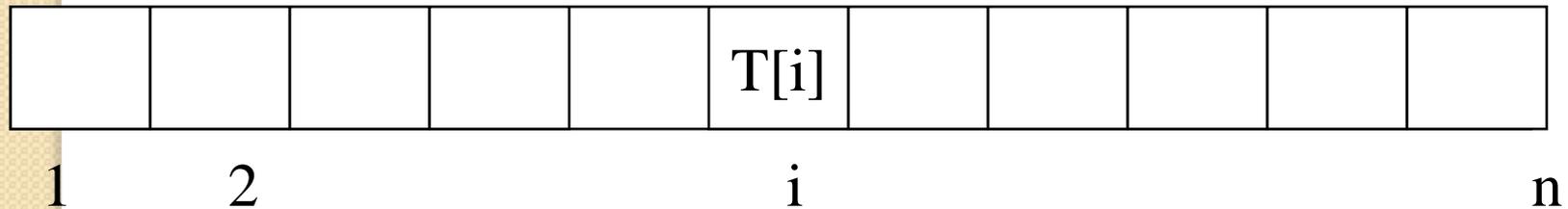
$i:=i+1;$

fintantque

Recherche séquentielle d'un élément e dans un tableau trié T

$$T[1..i-1] < e$$

$$e \leq T[i..n]$$



```
i:=1;  
Tantque i<=n et T[i]<e faire  
    i:=i+1;  
fintantque  
{(i=n+1) ou T[i]>=e}
```

■ Complexité :

– au mieux : 1

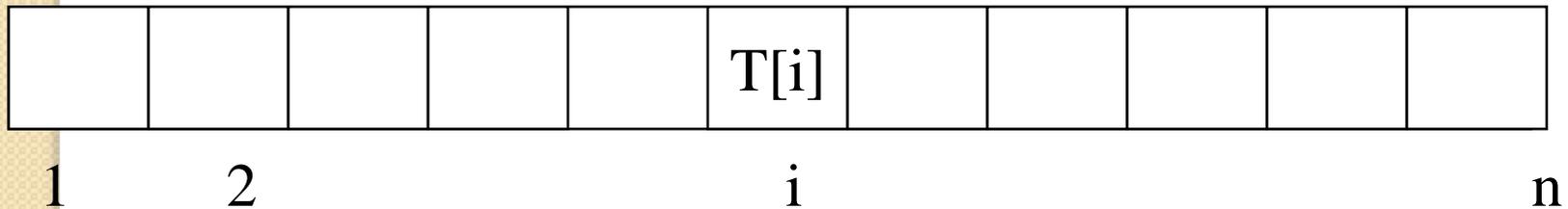
– au pire : n

– en moyenne : $n/2$

(que e appartienne ou non
à T)

Recherche séquentielle d'un élément e dans un tableau T non trié

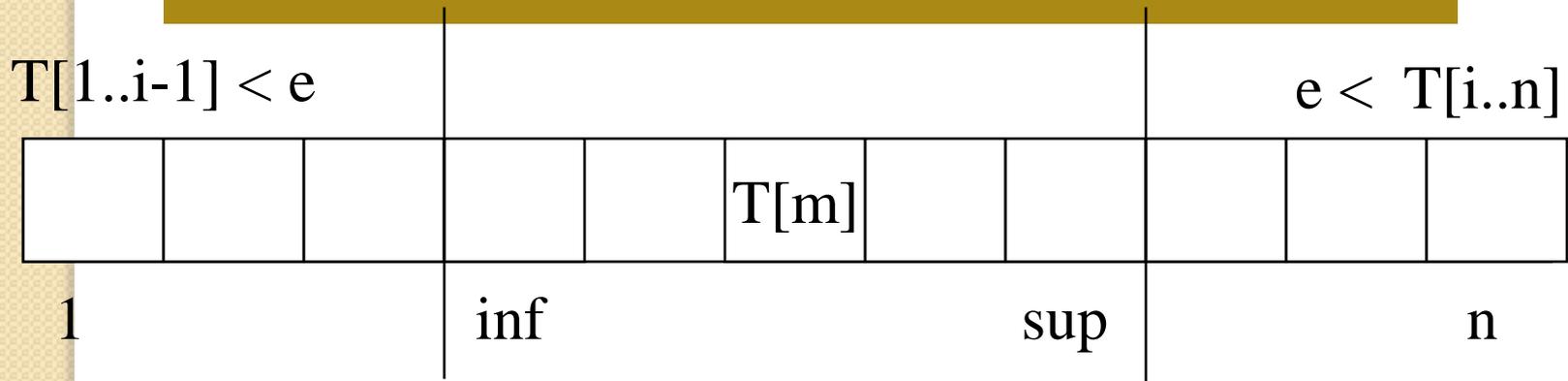
$T[1..i-1] \neq e$



```
i:=1;  
Tantque i<=n et T[i]≠e  
Faire i:=i+1;  
Fintantque  
{(i=n+1) ou T[i]=e}
```

- Complexité :
 - au mieux : 1
 - au pire : n
 - en moyenne : $n/2$ (si e appartient à T)

Recherche dichotomique d'un élément e dans un tableau trié T



```
trouvé:=faux; inf:=1; sup:=n;
Tantque inf<=sup et non trouvé faire
    m:=(inf+sup) div 2;
    si T[m]=e alors trouvé:=vrai
    sinon
        si T[m]<e alors inf:=m+1
        sinon sup:=m-1
    finsi
finsi
fintantque
```

- Complexité :
 - au mieux : 1
 - au pire : $\log_2 n$

Insertion d'un élément dans un tableau non trié

- A priori, on ajoute le nouvel élément en fin de tableau
- S'il existe un critère définissant l'endroit d'insertion, on est dans un cas comparable au tableau trié

Insertion d'un élément dans un tableau trié

- Chercher la place de l'élément à insérer
- Effectuer l'insertion

Méthode séquentielle de recherche de la position d'insertion dans un tableau trié

fonction position($T[1..n], e$:élément):entier,

p, i :entier;

début

si $T[1] > e$ alors $p := 1$

sinon

$i := n$;

tantque $T[i] > e$ faire $i := i - 1$ fintantque

$p := i + 1$;

finsi

retourner(p);

fin

■ Complexité :

– au mieux : 1

– au pire : n

– en moyenne : $n/2$

Méthode séquentielle de recherche de la position d'insertion dans un tableau trié (variante)

fonction position($T[1..n], e$:élément):entier,

p, i :entier;

Début

$i := N$;

tantque $i > 0$ et $T[i] > e$

faire $i := i - 1$

fantantque;

$p := i + 1$;

retourner(p);

fin

■ Complexité :

– au mieux : 1

– au pire : n

– en moyenne : $n/2$

Attention si $i=0$,
il ne faut pas que $T[i]$ soit testé
=> Dépend du compilateur

Méthode dichotomique de recherche de la position d'insertion dans un tableau trié

fonction position(T[1..n]:tableau d'éléments,e:élément):entier,

p,inf,sup,m:entier;

début

si T[n]≤e alors

p:=n+1

sinon

inf:=1; sup:=n;

tantque inf < sup faire

m:=(inf + sup) div 2;

si T[m]≤e alors inf:= m+1 sinon sup:=m finsi

fintantque

p:=sup;

finsi

retourner(p);

fin

■ Complexité :

– au mieux : 1

– au pire : $\log_2 n$

Insertion d'un élément e connaissant la position p souhaitée

début

pour $i:=n$ à p faire
 $T[i+1]:=T[i]$

finpour;

$n:=n+1$;

$T[p]:=e$;

fin

- Complexité :
 - au mieux : 1
 - au pire : n
 - en moyenne : $n/2$

Suppression d'un élément dans un tableau trié

- Rechercher une occurrence de la valeur e à supprimer
- Si on la trouve, « tasser » les éléments du tableau

Remarques sur les tableaux

- Ce sont des structures séquentielles mais qui permettent aussi un accès direct à partir du numéro de la place (accès par position)
- Efficaces en recherche s'ils sont triés, en insertion s'ils ne sont pas triés

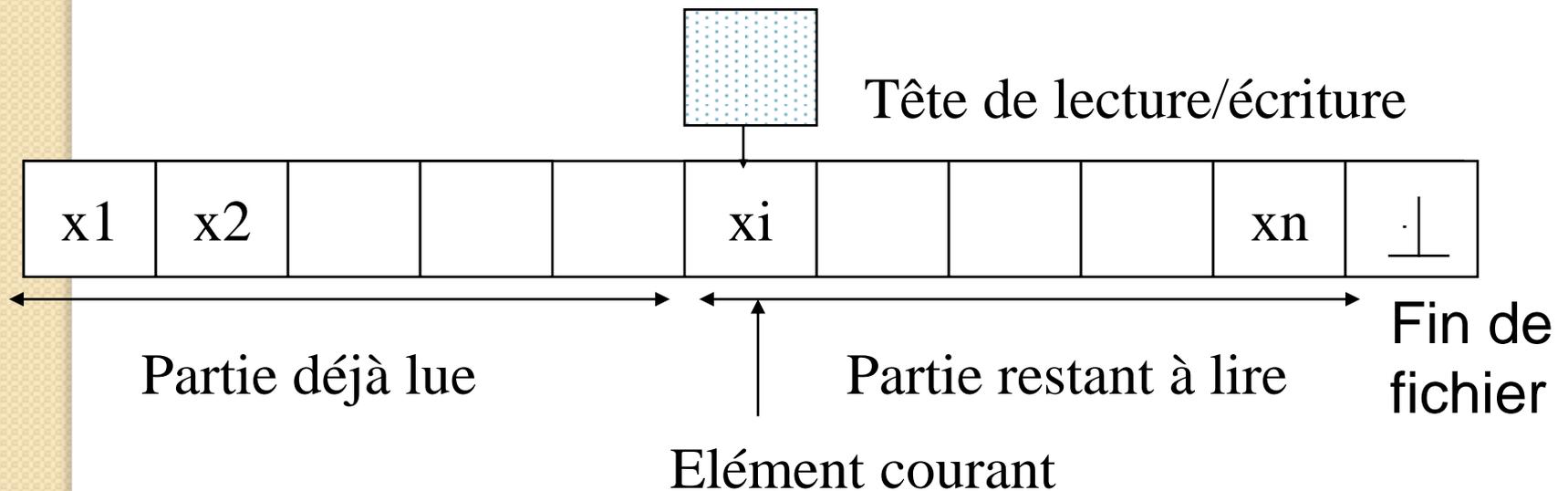
4. Le fichier séquentiel

- Suite finie d'éléments d'un même type
- Exemples :
 - fichier des livres d'une bibliothèque
 - enregistrements sur un magnétophone
 - ensemble des passagers attendant la montée dans l'avion

Caractéristiques d'un fichier séquentiel

- Ensemble de places totalement ordonné
- Chaque place est occupée par un élément d'un même ensemble
- On peut détecter la fin de fichier
- On peut définir une correspondance entre l'ordre dans le fichier et l'ensemble des éléments classés dans le fichier : il y a une règle de rangement

Principe de parcours d'un fichier séquentiel



Primitives d'accès (I)

- Fin de fichier **eof(fichier)**
 - vrai si la tête de lecture/écriture est en face de la marque de fin de fichier
- Accès au premier élément **ouvrir(fichier)**
 - positionne la tête de lecture/écriture sur le premier élément du fichier ou sur la marque de fin de fichier s'il est vide
- Accès à l'élément courant **lire(fichier,val)**
 - copie l'élément courant dans val et avance la tête d'une position

Primitives d'accès (2)

- **créer(fichier)**
 - crée un fichier vide (ou détruit le contenu)
- **écrire(fichier,val) :**
 - recopie en fin de fichier la variable val
 - avance le ruban d'une position (sur la fin de fichier),
 - possible si l'on est en fin de fichier uniquement
- association entre un nom de fichier physique et un identificateur de fichier interne au programme

Somme des éléments d'un fichier f non vide de nb entiers

```
s,c:entier;  
début  
s:=0;  
ouvrir(fichier);  
lire(fichier,s);  
tantque non eof(fichier) faire  
    lire(fichier,c);  
    s:=s+c;  
fintantque  
retourner(s)  
fin
```

- Complexité :
 - taille du fichier : n

Accès au k-ième élément d'un fichier séquentiel

```
Procédure acces(fichier in, k in, trouvé:booléen out, val out);  
i:entier; c:fiche; trouvé:booléen;  
début  
ouvrir(fichier);  
trouvé:=faux;  
i:=1;  
tantque i<k et non eof(fichier) faire  
    lire(fichier,c);  
    i:=i+1;  
Fintantque;  
si non eof(fichier) alors trouvé:=vrai; lire(fichier,val); finsi  
fin
```

Accès associatif à un élément du fichier

c:fiche; trouvé:booléen; { recherche de la valeur val }

début

ouvrir(fichier);

 si eof(fichier)

 alors trouvé:=faux

 sinon

 lire(fichier,c);

 tantque non eof(fichier) et (c<>val) faire lire(fichier,c);

 fintantque

 trouvé:=c=val;

 finsi

fin

Éclatement d'un fichier en deux

- On suppose que le fichier f comprend des entiers naturels
- On construit deux fichiers :
 - l'un f_{pair} contenant les nbs pairs
 - et l'autre f_{impair} les nombres impairs

Éclatement d'un fichier en deux

Début

```
ouvrir(f); créer(fpair); créer(fimpair);
tantque non eof(f) faire
    lire(f,c);
    si c mod 2 = 0
        alors écrire(fpair,c);
        sinon écrire(fimpair;c);
    finsi;
fintantque;
```

Fin;

Insertion d'un élément à la place k dans un fichier f

- On construit un fichier g par concaténation de :
 - $(k-1)$ premiers éléments de f
 - l'élément à ajouter
 - $(n-k+1)$ derniers éléments de f

Insertion d'un élément e à la place k dans un fichier f

Début

```
ouvrir(f); créer(g); i:=1; possible:=faux;  
tantque non eof(f) et (i<k) faire lire(f,c); écrire(g,c);i:=i+1;  
fintantque;  
si i=k  
    alors  
        écrire(g,e);  
        tantque non eof(f) faire lire(f,c); écrire(g,c); fintantque;  
        possible:=vrai;  
    sinon  
        possible:=faux;  
finsi
```

fin

Remarques sur les fichiers séquentiels

- Tous les accès sont séquentiels
- Le coût de l'algorithme est proportionnel à la taille du fichier
- La mise à jour nécessite (presque toujours) la construction d'un nouveau fichier