Algorithmique & programmation

Chapitre 3 : Fichiers séquentiels

Algorithme traitant un seul fichier

Accès associatif non trié

Accès associatif (fichier non trié)

- On souhaite écrire la procédure suivante

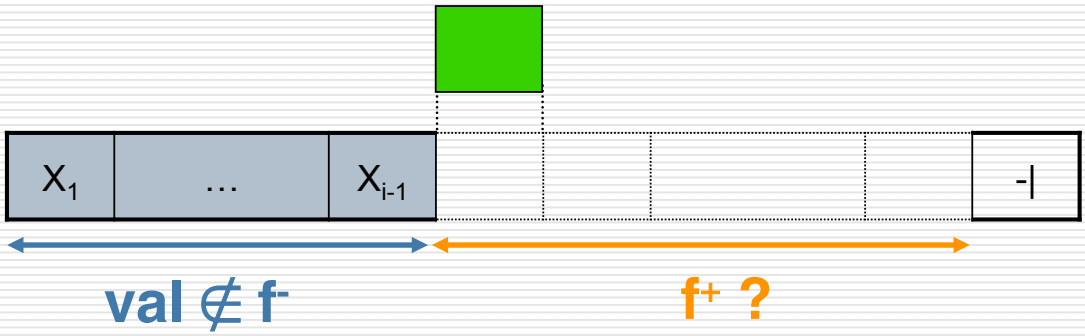
procédure accès (d f : fichier de t ; d val : t ;

r trouvé : booléen) ;

spécification $\{\ \}$ ➔ $\{(trouvé\ ,\ val \in f)$ $\vee\ (\neg trouvé\ ,\ val \notin f)\}$

Accès associatif (fichier non trié)

- Esquisse d'un raisonnement "habituel"
  - À un instant donné, on n'a toujours pas rencontré val dans le sous-fichier déjà examiné
  - En schématisant



Accès associatif (fichier non trié)

- Raisonnement par récurrence (esquisse)

Hypothèse val ∉ f⁻

- $f^+ = <>$ ⇨ *val* ∉ *f* ⇨ trouver := faux ; ✳
- $f^+ \neq <>$ lire (f , c) ;
  - ⇨ *val* ∉ f^-, *c* = *Df*
    - c = val ⇨ *val* ∈ *f* ⇨ trouver := vrai ; ✳
    - c ≠ val ~~⇨ *val* ∉ *f*~~ ➡ ***H***

Trouver l'itération ?

Chemin depuis le retour à l'hypothèse vers les conditions

Accès associatif (fichier non trié)

- On ne peut pas écrire

  Itération tantque (non fdf(f)) **et** (c ≠ val) faire …

- Une seule solution
  - Utiliser un booléen qui indique que **val** a été trouvé
    - ➔ ajouter le booléen dans l'hypothèse

Accès associatif (fichier non trié)

- Raisonnement par récurrence

Hypothèse val ∉ f⁻, ¬ égal

➤ $f^+ = <>$ ⇨ *val ∉ f* ✳

➤ $f^+ \neq <>$ lire (f , c) ;

⇨ *val ∉ f⁻, c = Df⁻*

➤➤ c = val ⇨ *val ∈ f* ⇨ égal := vrai ; ✳

➤➤ c ≠ val ⇨ *val ∉ f* ➡ ***H***

Itération tantque non fdf(f) et non égal faire...

Initialisation relire (f) ; égal := faux ; ➡ ***H***

Accès associatif (non trié, calcul du résultat)

- Tableau de sortie

fdf(f)	égal	résultat
vrai	vrai	trouvé := vrai (val = Df^-)
vrai	faux	trouvé := faux
faux	vrai	trouvé := vrai
faux	faux	impossible (tantque)

- trouvé est vrai lorsque égal est vrai
  - **trouvé := égal ;**

procédure **accès1**

procédure accès (d f : fichier de t ; d val : t ; r trouvé : booléen) ;
spécification { } ➔ *{(trouvé , val ∈f) ∨ (¬trouvé , val ∉f)}*
    c : t ; égal : booléen ;
debproc
    relire (f) ; égal := faux ;
    {val ∉f , (égal, val ∈f) ∨ (¬égal, val ∉f)}
    tantque non fdf (f) **et non** égal **faire**
        lire (f , c) ;
        {val ∉f^- , c = Df , ¬ égal }
        si c = val **alors**
            égal := vrai ;
            {val ∉f^- , c = val , égal , val ∈f}
        finsi ;
        {val ∉f^-, (égal , val ∈f) ∨ (¬égal, val ∉f)}
    finfaire ;
    {fdf (f) ou égal}
    trouvé := égal ;
finproc ;

procédures **acces** (fichier d'entiers)

```
-- Importer le module générique Sequential_io, use inutile
with Sequential_Io;

package P_Fentier is
   -- Instancier le paquetage Sequential_Io
   -- Pour manipuler des fichiers de Integer
   package P_Entier_Io is new Sequential_Io(Integer);
   use P_Entier_Io;

   -- Première version de la procédure
   procedure acces1(F : in P_Entier_Io.File_Type ; val : in
     integer ; trouve : out boolean);

   -- Seconde version de la procédure
   procedure acces2(F : in P_Entier_Io.File_Type ; val : in
     integer ; trouve : out boolean);

end P_Fentier;
```


procédure **acces1**

```
procedure acces1(F : in P_Entier_Io.File_Type ; val : in
    integer ; trouve : out boolean) is

--spec { } ➔ {(trouvé , val ∈ f) ∨ (¬trouvé , val ∉ f)}

   c : integer ;    egal : boolean ;
begin
   reset(F) ;          --on se positionne au début du fichier
   egal := false ;   --initialisation de egal à faux
   while not End_Of_File(F) and not egal loop
      read (F , c) ;
      if c = val then
         egal := true ;
      end if ;
   end loop ;
   trouve := egal ;
end acces1 ;
```

Accès associatif (autre version)

- On peut se passer du booléen en changeant d'hypothèse
  - Il faut qu'une valeur non encore testée soit disponible dans l'hypothèse
    - La seule valeur que l'on peut avoir disponible est : **Df^-**
  - Si **Df^-** est disponible alors c'est que $val \notin f^{--}$
- On propose donc la nouvelle hypothèse

$$\mathbf{val \notin f^{--}\ ,\ c = Df^-}$$

  - *Étudions l'impact de ce changement*

Accès associatif (autre version)

- Raisonnement par récurrence

Hypothèse $val \notin f^{--}$, $c = Df^-$

➤ $c = val$ ⇨ $val \in f$ ⇨ trouvé := vrai ; ✳

➤ $c \neq val$

➤➤ $f^+ = <>$ ⇨ $val \notin f$ ⇨ trouvé := faux ; ✳

➤➤ $f^+ \neq <>$ ⇨ lire (f , c) ; ➡ ***H***

Itération tantque ($c \neq val$) et non fdf(f) faire...

Initialisation relire (f) ;

➤ $f^+ = <>$ ⇨ trouvé := faux ; ✳

➤ $f^+ \neq <>$ ⇨ lire (f , c) ; ➡ ***H***

Accès associatif (autre version)

- Tableau de sortie

fdf(f)	c = val	résultat
vrai	vrai	trouvé := vrai (val = Df^-)
vrai	faux	trouvé := faux
faux	vrai	trouvé := vrai
faux	faux	impossible (tantque)

- trouvé est vrai lorsque c = val
  - **trouvé := c = val ;**

procédure **accès2** (autre version)

procédure accès2 (d f : fichier de t ; d val : t ; r trouvé : booléen) ;
spécification { } ➔ *{(trouvé , val ∈f) ∨ (¬trouvé , val ∉f)}*
    c : t ;
debproc
    relire (f) ;
    si fdf(f) **alors**
        trouvé := faux ;
    sinon
        lire(f, c) ; *{val ∉f^-, c = Df}*
        tantque (c ≠ val) **et non fdf** (f) **faire**
            {val ∉f^-}
            lire (f , c) ;
            {val ∉f, c = Df}
        finfaire ;
        {fdf (f) ou (c = val) , val ∉f, c = Df}}
        trouvé := c = val ;
    finsi ;
finproc ;

procédure **acces2**

```
procedure acces2(F : in P_Entier_Io.File_Type ;
                 val : in integer ; trouve : out boolean) is
--spec { } ➔ {(trouvé , val ∈ f) ∨ (¬trouvé , val ∉ f)}
  c : integer ;   egal : boolean ;
begin
  reset(F) ;      --on se positionne au début du fichier
  if End_Of_File(F) then  --traitement du fichier vide
    trouve := false ;
  else
    --traitement d'un fichier contenant au moins un élément
    read (F , c) ;
    while c <> val and not End_Of_File(F) loop
      read (F , c) ;
    end loop ;
    trouve := c=val ;
  end if ;
end acces2 ;
```