CHAP V FICHIERS ET ALGORITHMIQUE

Structure homogène dont les différents éléments résident sur un support externe (disque magnétique, CD-ROM). Cette structure a une organisation sur le support externe.

Les organisations les plus courantes sont :

séquentielle, directe, séquentielle indexée.

D'autres: séquentielle chaînée, inversée, arborescente.

Fichier # Base de Données

Base de données = structure hétérogène avec son organisation:

B. D. hiérarchique,

B. D. en réseau,

B. D. relationnelle

Les programmes écrits jusqu'ici lisaient tous l'entrée standard **stdin** et écrivaient tous sur la sortie standard **stdout** , qui sont toutes les deux définies automatiquement, pour un programme donné, par le système d'exploitation sur lequel on travaille (cf l 'option UNIX) .

On s'interesse maintenant à l'écriture d'un programme qui accède à un fichier qui n'est pas encore connecté au programme.

Pour cela il faut pouvoir ouvrir le fichier en passant par un pointeur de fichier. On pourra alors lire ou écrire . Enfin on fermera le fichier.

I FICHIERS SÉQUENTIELS

Représentation sur support externe d'une liste de données de même type. Accès séquentiel.

En C, un fichier est déclaré par une variable de type pointeur de fichier qui pointe sur une structure contenant des informations sur le fichier, telles que l'adresse du tampon, la position du caractère courant dans le tampon, des indicateurs permettant de savoir si le fichier est ouvert en lecture ou en écriture, si des erreurs sont intervenues ou si la fin du fichier est atteinte.

Pour faire la déclaration on utilise le type FILE

FILE * <identificateur>

Les opérations sur un fichier s'effectuent à partir de fonctions d'ouverture (*fopen* renvoie un pointeur) puis de lecture et écriture :

FILE *fopen (char *nom , char *mode) ;

Par exemple : ***FILE *fp ;***

fp = fopen ("essai" , "w") ;

On déclare la variable ***fp*** puis on ouvre le fichier 'essai' en écriture ('*w*'). Le 1er argument est le nom du fichier et le suivant précise le modes d'ouverture qui peut être la **lecture ("r"), l'écriture ("w"), et l'ajout ("a").**

Si l'on ouvre un fichier qui n'existe pas en écriture ou en ajout, il est créé. Ouvrir un fichier existant en écriture écrase le contenu précédent alors que l'ouvrir en ajout conserve ce contenu. Il y a erreur si on veut lire un fichier qui n'existe pas : *fopen* renvoie alors ***NULL***.

Il faut pouvoir lire ou écrire dans le fichier après son ouverture en utilisant des fonctions standard de <stdio.h> avec les prototypes

```
int fgetc ( FILE * );
char * fgets ( char * , int , FILE * );
int fputc ( int , FILE * );
int fputs ( char * , FILE * );
int fscanf ( FILE * , char *, <type> *, ... );
int fprintf ( FILE *, char *, <type> , ... );
```

Tester la fin du fichier ***int feof (FILE *);***

Fermer le fichier ***int fclose (FILE *);***

Une constante particulière (caractère ou entier) est prédéfinie ***EOF***: elle peut être renvoyée par ***fgetc, fputc, fputc, fputs, fscanf, getc, getchar, putc, putchar, ...***

II ALGORITHMES FONDAMENTAUX

1 seul fichier avec création, visualisation,recherche...

Algorithmes = fonctions ou programmes indépendants

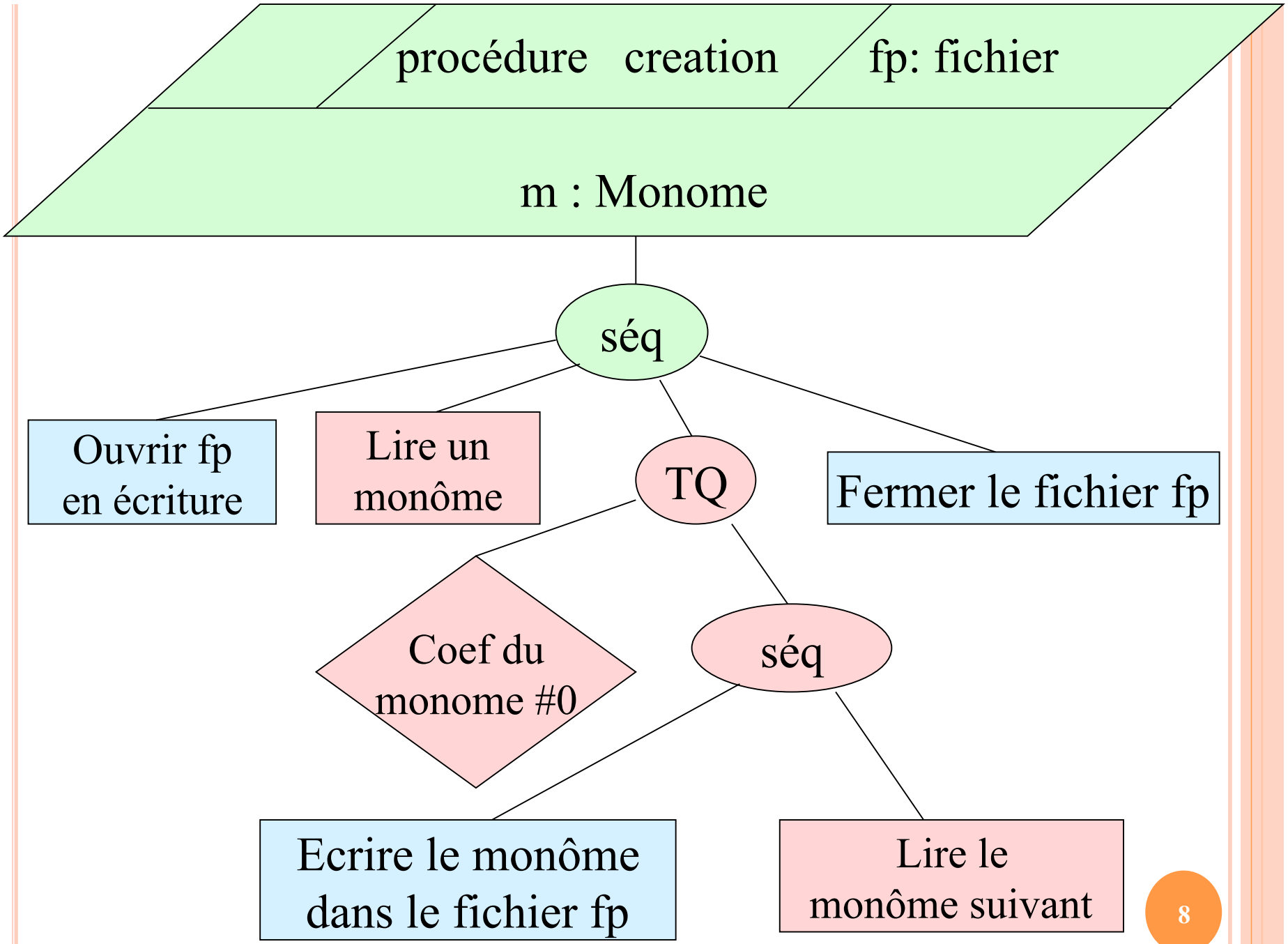
On supposera ici qu'on veut un fichier de monomes (appelé «fmonome») où chaque monome est composé d'un entier d et d'un réel c

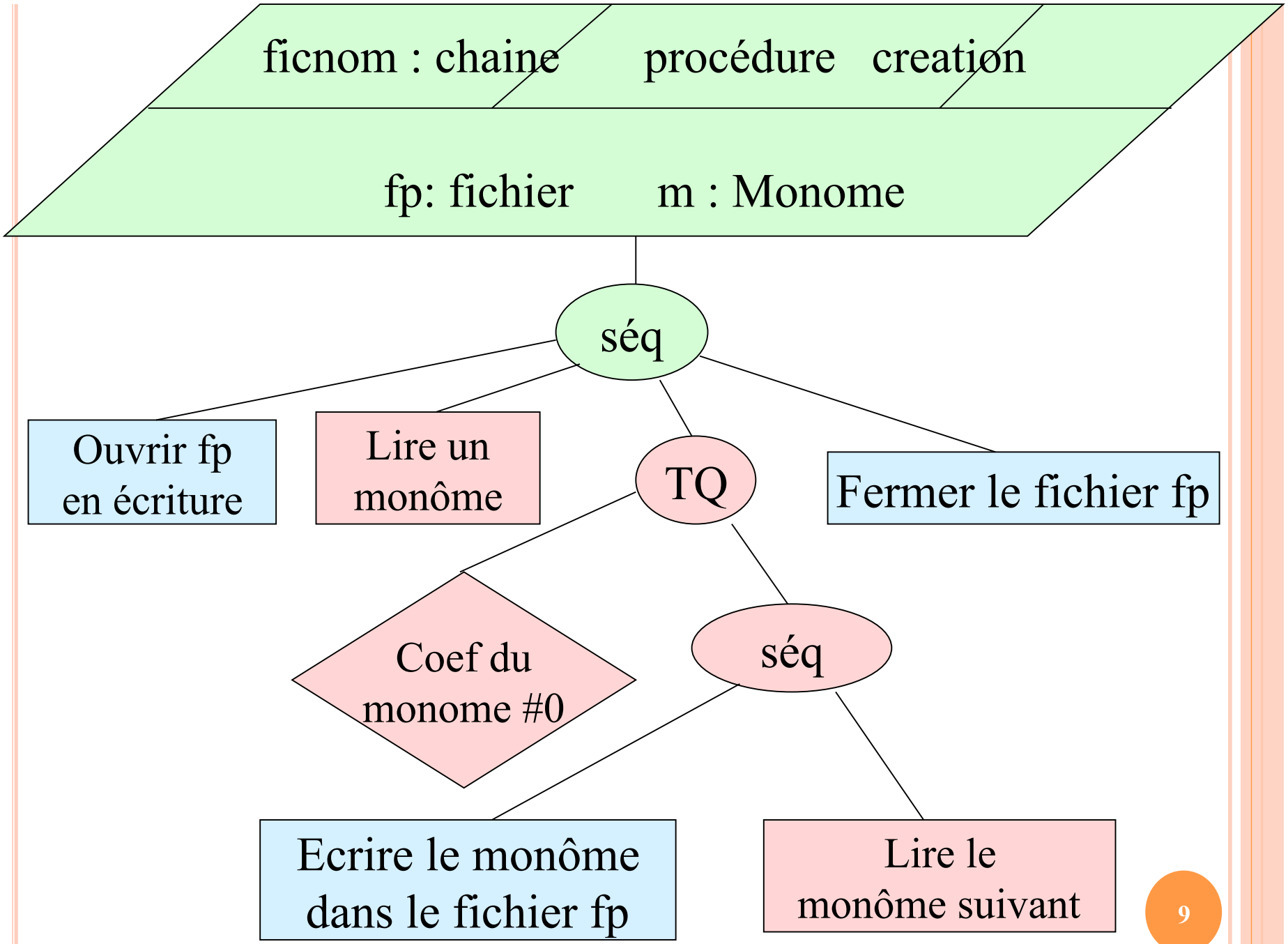
```
struct monome   {  int d ;
                   float  c ; }
      typedef   struct monome  Monome ;
```

```
typedef  struct    {  int d ;
                      float  c ; }  Monome ;
```

Création :ouvrir le fichier en écriture ;
lire un monôme ;
TQ le coeff c du monôme est de 0 FAIRE
déposer le monôme puis lire le suivant ;
fermer le fichier

```
void creation1  ( FILE  * fp )
    {    Monome m ;
     fp = fopen (" fmonome", " w " ) ;
     scanf  ("%f %d" , & m.c , & m.d ) ;
     while  ( m.c  != 0 )
         { fprintf ( fp , "%f  %d" , m.c , m.d ) ;
           scanf ("%f%d" , &m.c , &m.d ) ;    }
     fclose ( fp ) ;     }
```


procédure creation
fp: fichier
m : Monome
séq
Ouvrir fp
en écriture
Lire un
monôme
TQ
Fermer le fichier fp
Coef du
monome #0
séq
Ecrire le monôme
dans le fichier fp
Lire le
monôme suivant

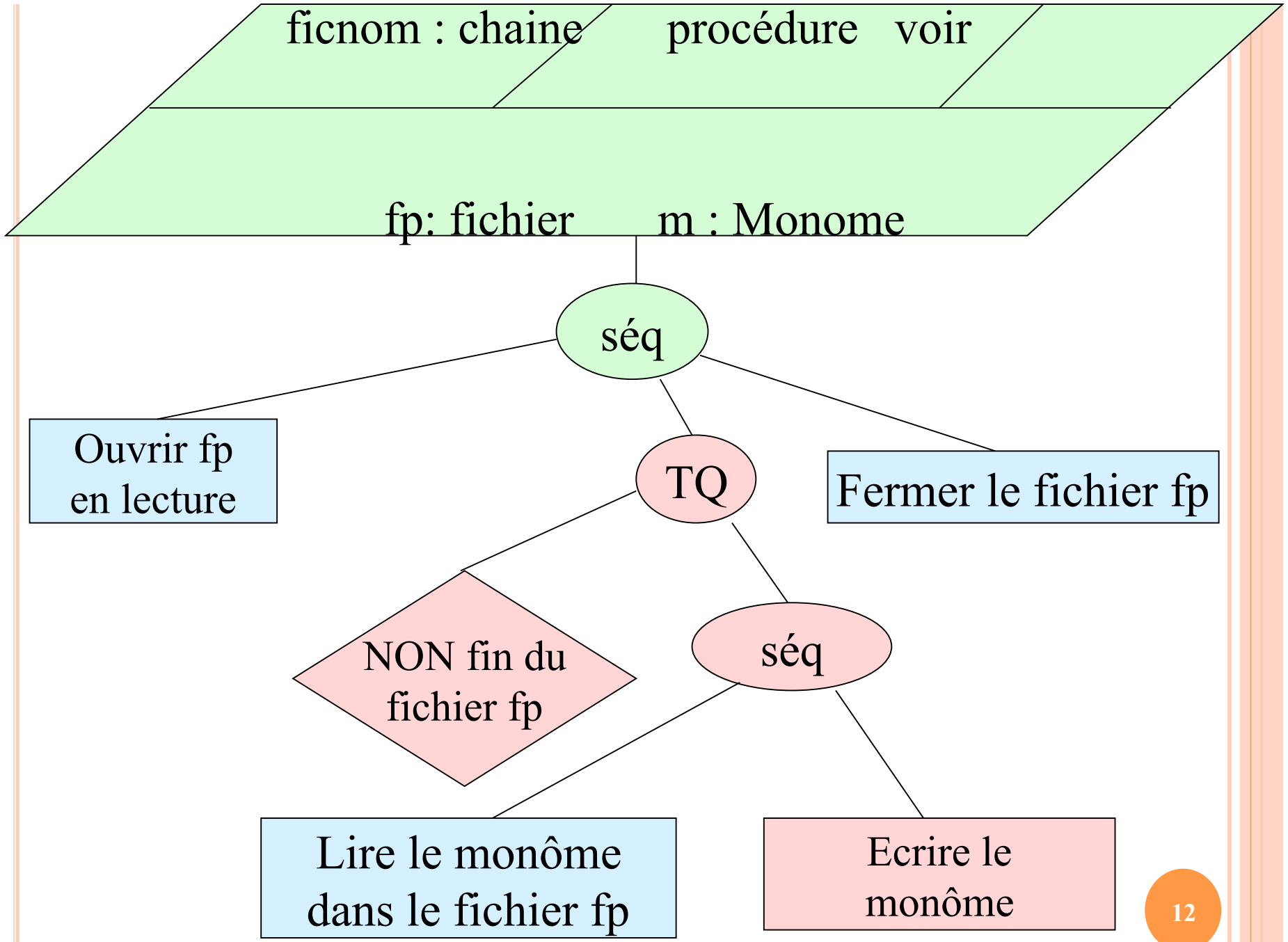

ficnom : chaine
procédure creation
fp: fichier
m : Monome
séq
Ouvrir fp
en écriture
Lire un
monôme
TQ
Fermer le fichier fp
Coef du
monome #0
séq
Ecrire le monôme
dans le fichier fp
Lire le
monôme suivant

```
void creation1  ( char   nom_fic [ ] )
  {  Monome m ;
     FILE  * fp ;
     fp = fopen ( nom_fic, " w " ) ;
     if (fp == NULL) printf (" erreur\n ");
     else {
          scanf  ("%f %d" , & m.c , & m.d ) ;
          while  ( m.c  != 0 )
               {
                 fprintf ( fp, "%f  %d" , m.c , m.d );
                      scanf ("%f%d",  &m.c,  &m.d);
     }
          fclose ( fp ) ;
          }
  }
```

Voir : ouvrir le fichier en lecture ;
TQ NON fin du fichier FAIRE
lire le monôme puis afficher le monôme ;
fermer le fichier .

```
void  voir1  (char nom_fic [ ])
{   Monome m ; FILE * fp = fopen ( nom_fic , "r" ) ;
     while ( ! feof ( fp ) )
     { fscanf(fp, "%f %d", &m.c , &m.d ) ;
      printf ("coef =%f  degré =%3d\n", m.c, m.d);}
     fclose ( fp ) ;
}
```

```
void  voir1  ( FILE  * fp )
{ Monome m ;
   fp = fopen (" fmonome ", " r " ) ;   /* ...  */ }
```


ficnom : chaine
procédure voir
fp: fichier
m : Monome
séq
Ouvrir fp
en lecture
TQ
Fermer le fichier fp
NON fin du
fichier fp
séq
Lire le monôme
dans le fichier fp
Ecrire le
monôme

```
main ( )
{ Monome m ; FILE * fp ;
  fp = fopen (" fmonome ", " r " ) ; /* ... */
```

```
void visurec ( FILE *fp )
    { Monome m ;
      if ( !feof ( fp ) )
            { fscanf ( fp, "%f %d", &m.c , &m.d ) ;
              printf ("coef =%f degré =%3d", m.c, m.d );
          visurec ( fp ) ; }
    }
```

```
void visu2 ( char fichier [ ])
    {   FILE * fp = fopen ( fichier , "r" ) ;
        visurec ( fp ) ;
       fclose ( fp ) ;
    }
```

Recherche: ouvrir le fichier en lecture ;

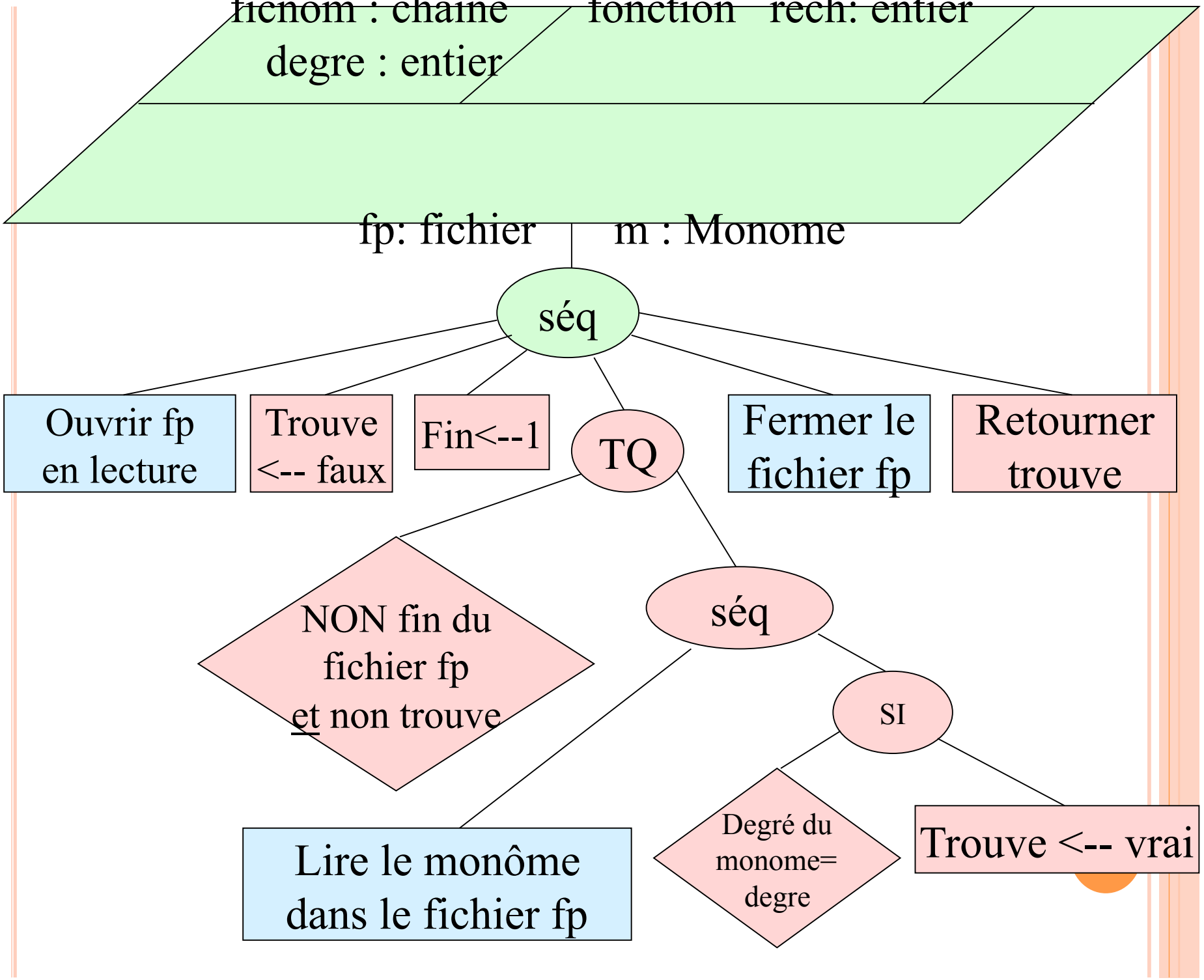
TQ non fin et non trouve faire

lire un monôme ;

comparer ce monôme avec l'info cherchée ;

fermer le fichier ;

```
int   rech  (char ficnom [], int degre )  /*renvoie 0 ou 1 */
     {Monome  m ; FILE * fp ;
      int trouve = 0,  fin = 1 ;
      fp = fopen  ( ficnom , « r ” ) ;
      while  (   fin != EOF  &&   ! trouve  )
            { fin= fscanf ( fp , «%f %d” , &m.c , &m.d );
              if ( degre == m.d )  trouve = 1 ;      }
      fclose ( fp ) ;  return trouve ;
     }
```


ficnom : chaine
degre : entier
fonction rech: entier
fp: fichier
m : Monome
séq
Ouvrir fp
en lecture
Trouve
<-- faux
Fin<--1
TQ
Fermer le
fichier fp
Retourner
trouve
NON fin du
fichier fp
et non trouve
séq
SI
Lire le monôme
dans le fichier fp
Degré du
monome=
degre
Trouve <-- vrai

```
Monome   rech2  (char nom [ ], int degre )
  {   FILE * f;                /*renvoie le monome*/
      Monome m;
      f = fopen  ( nom , "r" ) ;
       do
                  fscanf(f, "%f %d",&m.c, &m.d) ;
            while( ! feof ( fp)  &&   degre != m.d  );
      fclose ( f ) ;
      if  ( degre != m.d )    m.c = 0.0 ;
      return  m ;
  }
```

III MISE À JOUR D'UN FICHIER SÉQUENTIEL

- Modification ou ajout ou suppression.
- Accès séquentiel avec un pointeur de fichier qui avance à chaque lecture ou écriture. Il est donc **difficile de revenir en arrière** ... !
- Dans les Langages de Programmation un fichier est habituellement **ouvert** soit en **lecture** («r") soit en **écriture** (« w") soit parfois comme en C en **allongement** (« a") ou écriture à la fin. Il est donc **difficile de lire et écrire en même temps** .

Si fichier trié, il faut alors respecter l'ordre.

Une ou des mises à jour (modification, ajout, suppression). Les données vont être stockées dans un fichier intermédiaire de modifications. Si le fichier n'est pas trié, les ajouts peuvent s'effectuer en fin de fichier. Si le fichier est trié et pour les suppressions, il faut 2 fichiers : l'ancien et le nouveau.

```
void ajout  ( char fichier [ ])
{     Monome x ; FILE  * p;
      p = fopen  ( fichier , " a" ) ;
          scanf ("%f  %d" , &x.c ,&x.d ) )
      fprintf ( p , "%f%d " ,  x.c , x.d ) ;
      fclose  ( p ) ;
}
```

IV Algorithmes de base (2 fichiers séq.)

Algorithmes mettant en œuvre au moins 2 fichiers

- soit à cause d'une mise à jour,
- soit pour un traitement bien spécifique.

Fichier = suite séquentielle de données
et une variable étant de type pointeur sur FILE,
on peut faire correspondre à un fichier
un ensemble de données
avec les opérations habituelles sur un ensemble,
mais traitées séquentiellement

Soient 3 fichiers f1, f2 et f3 de même type

- Concaténation de 2 fichiers,
- Intersection de 2 fichiers
- Union de 2 fichiers,
- Différence de 2 fichiers
- Interclassement ou fusion de 2 fichiers
- Inclusion d'un fichier dans un autre
- Contenance d'un fichier dans un autre

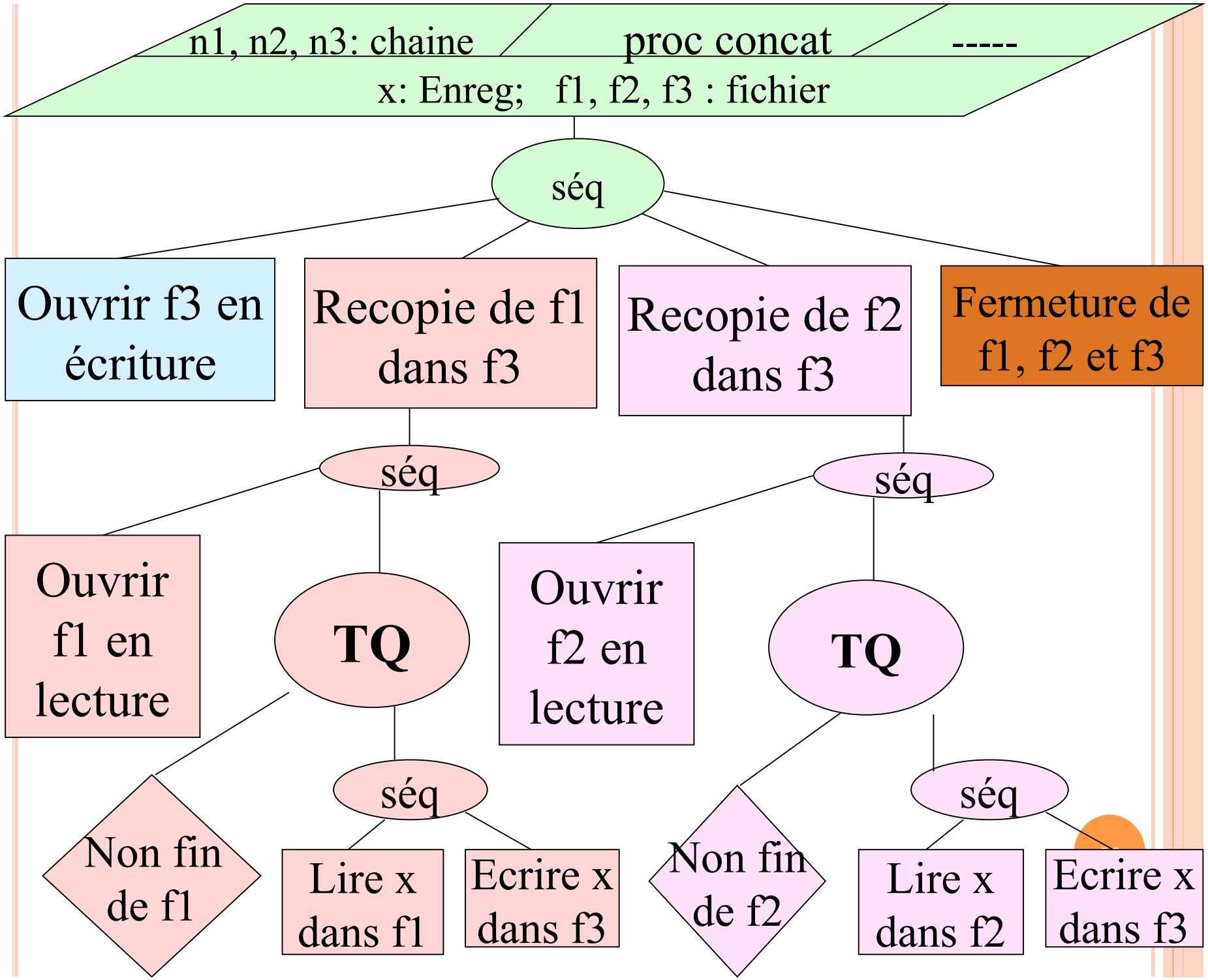
IV-1 CONCATÉNATION

- Concaténer 2 fichiers f1 et f2 consiste à les mettre bout à bout. Par exemple si **f1 = {a, c, f }** et **f2 = {d, g }** alors f3 = {a, c, f, d, g }

On recopie tout le fichier f1 dans le fichier f3 puis tout le fichier f2 dans f3 : f3 est ouvert en écriture et f1 et f2 sont ouverts en lecture.

Si la concaténation se fait dans l'un des 2 fichiers alors on ouvre ce fichier en allongement et on recopie l'autre dans celui-ci.

int liref (FILE *, Enreg *); /*renvoie 0 ou 1*/
void ecriref (FILE * , Enreg) ;


n1, n2, n3: chaine
proc concat
-----
x: Enreg; f1, f2, f3 : fichier
séq
Ouvrir f3 en écriture
Recopie de f1 dans f3
Recopie de f2 dans f3
Fermeture de f1, f2 et f3
séq
séq
Ouvrir f1 en lecture
TQ
Ouvrir f2 en lecture
TQ
séq
séq
Non fin de f1
Lire x dans f1
Ecrire x dans f3
Non fin de f2
Lire x dans f2
Ecrire x dans f3

```
void concat (char n1 [ ], char n2 [ ], char n3 [ ])
{ Enreg x ;   FILE * f1, * f2, * f3;
  f3 = fopen ( n3, "w") ;

/*      Recopie de f1  dans  f3                */
    f1 = fopen ( nom1, « r") ;
    while ( !feof(f1) )
            {   liref ( f1 , &x);
                ecriref ( f3, x );      }

   /*      Recopie  de  f2  dans   f3        */
      f2 = fopen (  nom2, « r" ) ;
      while ( ! feof (f2)  )
            { liref ( f2 , &x ) ; ecriref ( f3 , x ); }

 fclose ( f2 ) ; fclose ( f1 ) ;
  fclose ( f3 ) ;
}
```

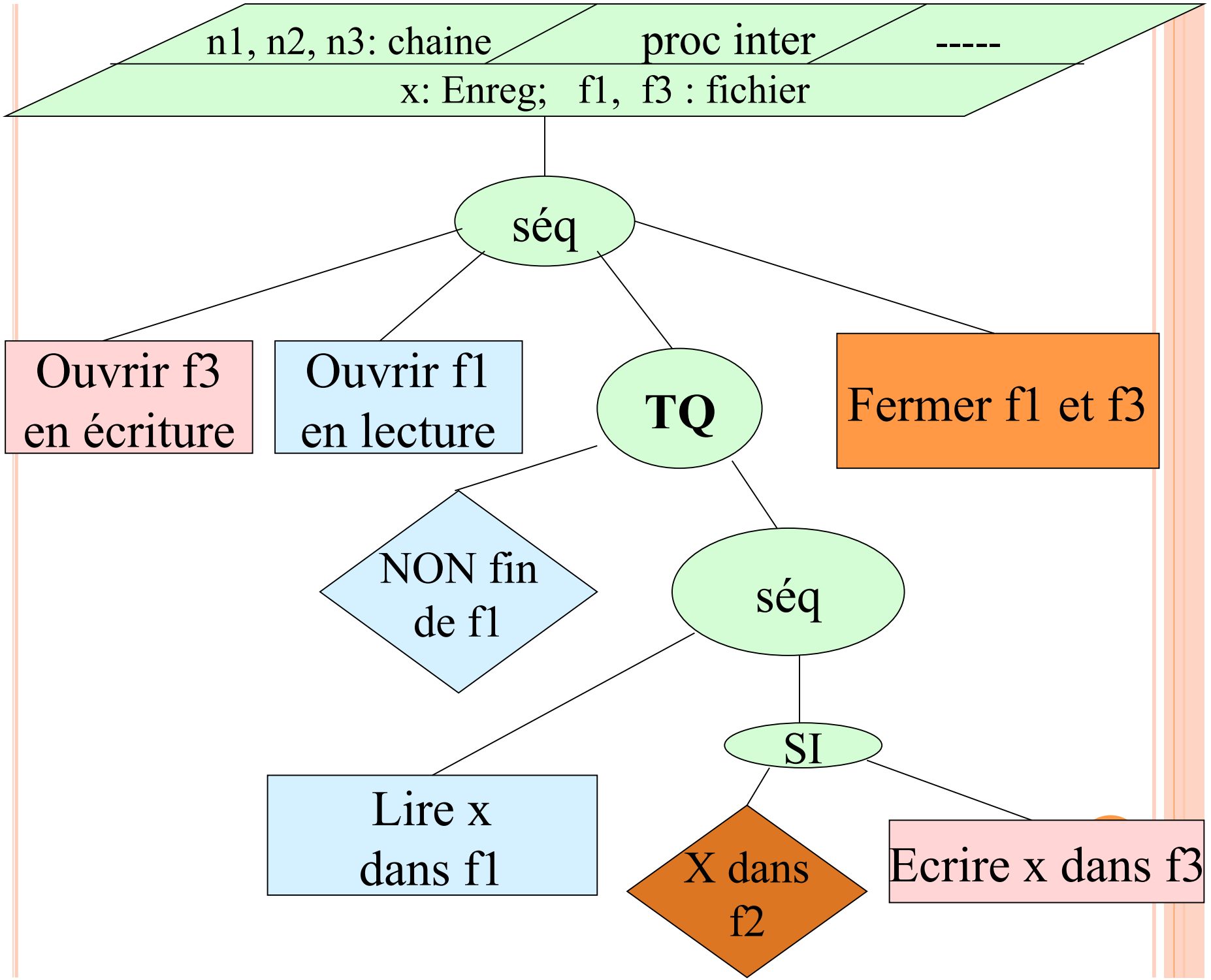
IV-2 INTERSECTION DE 2 FICHIERS

On ressort les éléments communs aux 2 fichiers. Par exemple si **f1 = {a, c, d, g, e }** et **f2= { b, g, a}** alors le résultat est un fichier f3 = {a, g}.

on parcourt un fichier complet, f1 par exemple . Pour chaque élément ou enregistrement du 1er fichier, on vérifie si celui-ci appartient aussi au 2ème fichier : si oui, on le recopie dans le 3ème fichier.

On énumère tout le 1er fichier f1et à chaque fois on énumère en partie le 2ème fichier f2 (recherche associative)

```
int  liref ( FILE *, Enreg *);  /*renvoie 0 ou 1*/
void ecriref ( FILE *, Enreg ) ;
int rech ( char [ ], Enreg );  /* renvoie 0 ou 1*/
```


n1, n2, n3: chaine
proc inter
-----
x: Enreg; f1, f3 : fichier
séq
Ouvrir f3 en écriture
Ouvrir f1 en lecture
TQ
Fermer f1 et f3
NON fin de f1
séq
SI
Lire x dans f1
X dans f2
Ecrire x dans f3

```
void inter (char nom1[], char nom2[], char
  nom3 [])
{ Enreg s ;
  FILE * f1;
  FILE * f3;
  f3 = fopen ( nom3, « w”) ;
  f1 = fopen ( nom1  , « r”) ;
  while  (! feof (f1) )
     {    liref ( f1 , &s )  );
          if ( rech ( nom2 , s ) )
                        ecriref ( f3 , s ) ;
     }
  fclose ( f1 ) ;
  fclose ( f3 ) ;
}
```

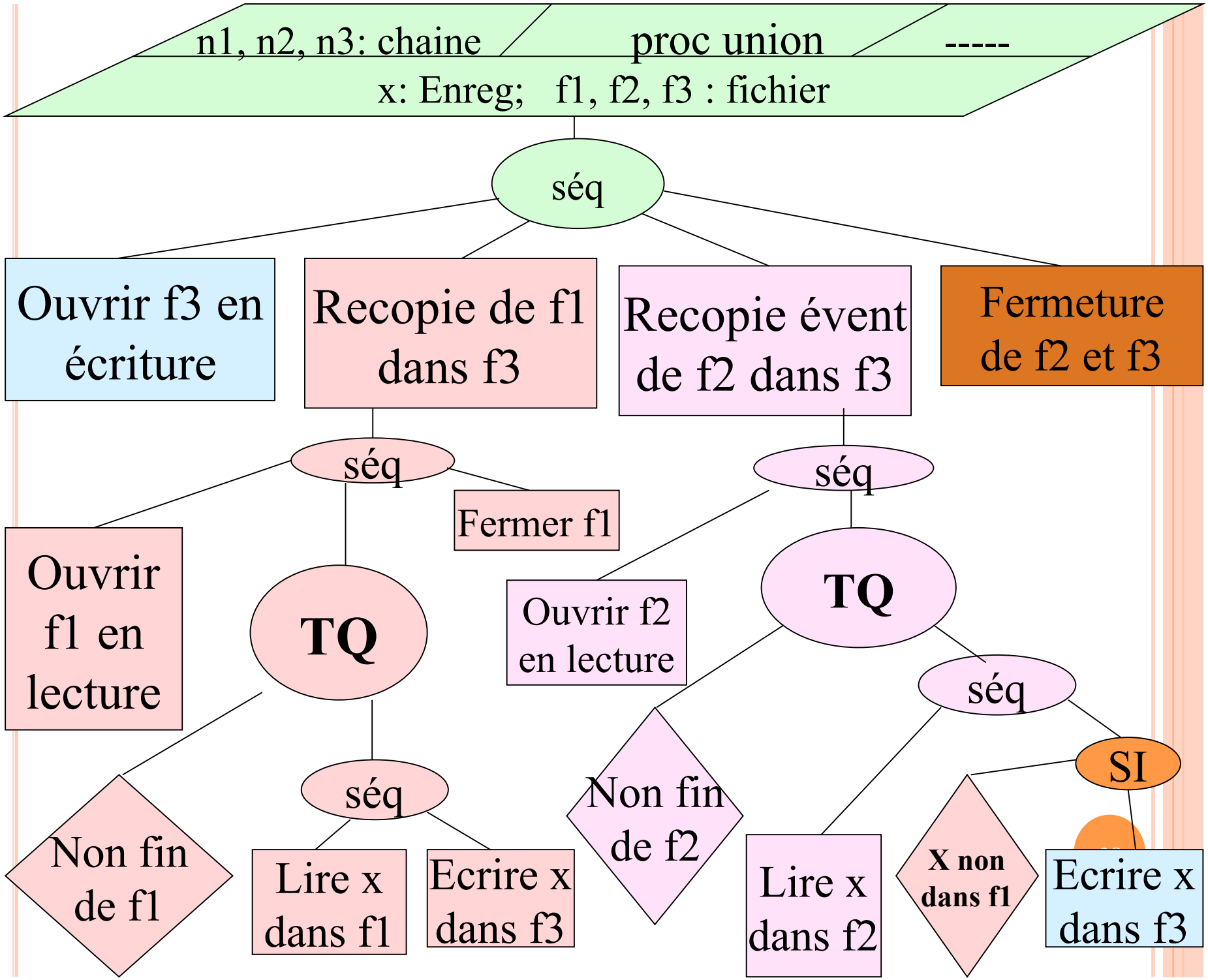

Nouvel opérateur → équivalent au champ de ce qui est pointé par

IV-3 UNION DE 2 FICHIERS

C'est l'union au sens mathématique et les éléments communs n'apparaissent qu'une fois . Par exemple si f1 = {c, a, d, f, g } et f2 = { a, g , f, e } alors le résultat est le fichier f3 = {c, a, d, f, g, e }

On recopie tous les éléments du 1er fichier f1 dans le fichier f3. Puis on recopie les éléments du 2ème fichier f2, qui n'existent pas dans le 1er, dans le 3ème fichier.

```
int   liref   ( FILE *,  Enreg   * );/* renvoie 0 ou 1   */
void ecriref ( FILE * , Enreg  ) ;
int rech ( char* , Enreg ) ;  /* 1 si trouvé sinon 0 */
```


n1, n2, n3: chaine
proc union
-----
x: Enreg; f1, f2, f3 : fichier
séq
Ouvrir f3 en écriture
Recopie de f1 dans f3
Recopie évent de f2 dans f3
Fermeture de f2 et f3
séq
séq
Ouvrir f1 en lecture
TQ
Fermer f1
Ouvrir f2 en lecture
TQ
Non fin de f1
séq
Non fin de f2
séq
SI
Lire x dans f1
Ecrire x dans f3
Lire x dans f2
X non dans f1
Ecrire x dans f3

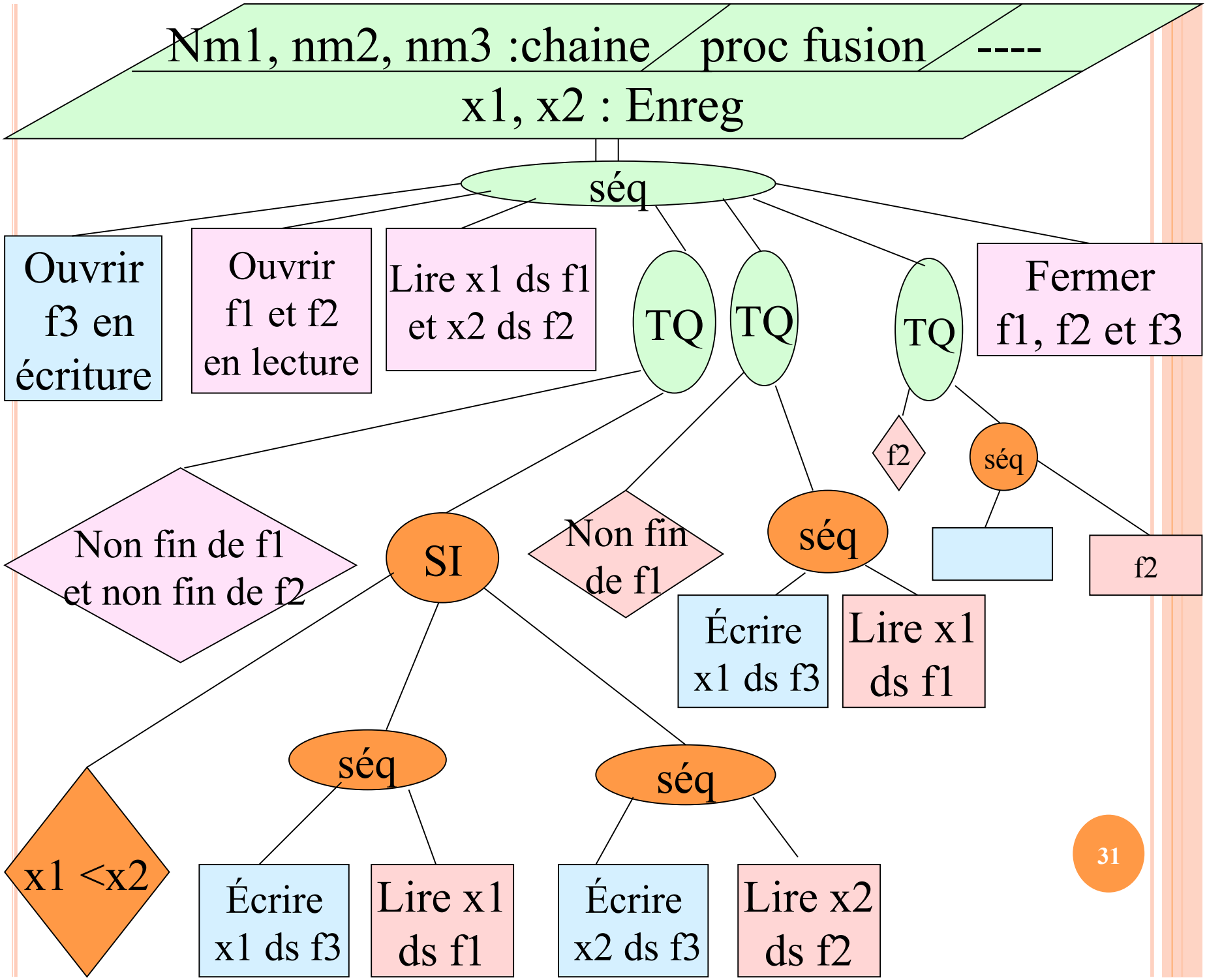
```
void funion (char nom1[], char nom2[], char
  n3[]))
{ Enreg x ; FILE * f1, * f2, * f3 ;
  f3 = fopen ( n3, « w”) ;
  /* Recopie de f1 dans f3       */
     f1 = fopen ( nom1 , « r”) ;
     while  ( !feof(f1) )
           { liref ( f1 , &x );  ecriref ( f3 , x ); }
     fclose ( f1 ) ;
  /* Recopie éventuelle de f2 dans f3 */
     f2 = fopen ( nom2 , « r”) ;
     while  ( !feof (f2) )
          { liref ( f2 , &x )  ;
              if ( ! rech ( nom1 , x ) )
                    ecriref ( f3 , x ) ;   }
  fclose ( f2 ) ;
  fclose ( f3 ) ;
}
```

IV-4 INTERCLASSEMENT OU FUSION DE 2 FICHIERS

C'est la fusion de 2 fichiers supposés ordonnés ou triés sur un critère identique pour créer un 3ème fichier respectant le même ordre. Par exemple si f1 = { a, d, e, h, k } et f2 = { b, e g } , alors f3 devient f3 = { a, b, d, e, e, g , h, k } .

L'algorithme se ramène à recopier tous les éléments des 2 fichiers f1 et f2 dans f3 en respectant l'ordre jusqu'à ce que l'un des 2 fichiers soit vide. Puis on recopie les éléments restant du fichier non terminé dans le fichier f3.

```
int   liref   (FILE *, Enreg * ); /*renvoie 0 ou 1*/
void ecriref ( FILE *, Enreg  ) ;
int comp ( Enreg , Enreg );/*renvoie 1 si< ou 0*/
```


Nm1, nm2, nm3 :chaine
proc fusion ----
x1, x2 : Enreg
séq
Ouvrir f3 en écriture
Ouvrir f1 et f2 en lecture
Lire x1 ds f1 et x2 ds f2
TQ
TQ
TQ
Fermer f1, f2 et f3
Non fin de f1 et non fin de f2
SI
Non fin de f1
séq
f2
séq
f2
Écrire x1 ds f3
Lire x1 ds f1
x1 <x2
séq
séq
Écrire x1 ds f3
Lire x1 ds f1
Écrire x2 ds f3
Lire x2 ds f2

```
void fusion (char nm1 [], char nm2 [], char
  nm3[])
{ Enreg   x1 , x2 ;
   FILE * f1; FILE * f2; FILE * f3;
   f3 = fopen (nm3, « w”) ;
   f1 = fopen ( nm1 , « r”); f2 = fopen ( nm2 , « r”) ;
   if (!feof (f1) ) liref ( f1 , &x1 ) ;
   if ( ! feof (f2 ) liref ( f2 , &x2 ) ;
   while  ( !feof ( f1 ) && ! feof ( f2 )  )
        if ( comp ( x1 , x2 ) )
          {   ecriref ( f3 , x1 ) ;  liref ( f1 , &x1 ) ;  }
             else  {     ecriref ( f3 , x2 );  liref ( f2 ,
   &x2 );  }
   while (  liref ( f1 , &x1 )  )  ecrire  ( f3 , x1 ) ;
   while (  liref ( f2 , &x2 )  )  ecrire  ( f3 , x2 ) ;
   fclose ( f1 ) ; fclose ( f2 ) ;    fclose ( f3 ) ;
}
```

```
void fusion (char nm1 [], char nm2 [], char nm3[])
{ Monome   x1 , x2 ;
   FILE * f1, * f2,* f3= fopen (nm3, « w ») ;
   f1 = fopen ( nm1 , « r »); f2 = fopen ( nm2 , « r ») ;
   if (!feof (f1) ) fscanf (f1, «%f%d », &x1.c, &x1.d);
   if (!feof (f2) ) fscanf (f2, «%f%d», &x2.c, &x2.d);
   while  ( !feof ( f1 ) && ! feof ( f2 )  )
        if ( x1.d < x2.d ){ fprintf (f3,«%f%d»,x1.c,x1.d);
   fscanf (f1, «%f%d », &x1.c, &x1.d);  }
                    else { fprintf (f3,«%f  %d  »,x2.c,  x2.d);
      fscanf (f2, «%f%d », &x2.c, &x2.d);}
   while (!feof(f1)) { fscanf (f1,«%f%d»,&x1.c,&x1.d);
                  fprintf (f3,«%f  %d »,x1.c, x1.d); }
    while (feof(f2) { fscanf (f2,«%f%d»,&x2.c,&x2.d);
                  fprintf (f3,«%f  %d »,x2.c, x2.d); }
   fclose ( f1 );     fclose ( f2 ); fclose ( f3 );        }
```