Fichiers

- structure composée de données dont le nombre n'est pas connu a priori et qui réside en mémoire secondaire (disques, CDROM, DVD, bandes magnétiques, ...)
- 3 types d'accès :
  - séquentiel → le fichier est parcouru systématiquement depuis le début jusqu'à l'élément recherché
  - direct → la position de l'élément recherché est fournie
  - selon une clé → chaque élément est associé à une clé. La recherche de la clé conduit à l'élément

Fichiers textes/ fichiers binaires

• Un fichier texte est formé de caractères ASCII, organisé en lignes, chacune
terminée par un caractère de contrôle de fin de ligne. Un caractère de fin de fichier termine le fichier. Toute tentative d'accès au delà de la fin du fichier déclenche l'exception `end_error`.
• Les fichiers texte peuvent être édités avec des éditeurs de texte et affichés de manière lisible à l'écran
• Un fichier binaire contient des données non textuelles. Ils ne prennent sens que s'ils sont traités par un programme adapté.
• Exemples de fichiers binaires : code exécutable d'un programme, fichiers son, video, etc...

Fichier texte Ada

- En Ada, le traitement des fichiers texte s'effectue grâce au paquetage prédéfini `Ada.Text_io`. Ce sont des fichiers séquentiels.

- On distingue les notions de fichier, variable déclarée du type `File_type`, et de fichier externe, objet extérieur au programme conservé en mémoire secondaire.

- Exemple :

```
procedure copier( source:in File_type;
                  destination:out File_type );
```

Création de fichiers texte

- Créer ou ouvrir un fichier consiste :
  - à associer une variable fichier à un fichier externe
  - à choisir un mode d'accès (lecture, écriture ou adjonction en fin de fichier)
- Le mode de lecture est spécifié par le type énumératif `File_Mode` dont les valeurs sont : `in_file,out_file,append_file`
- Le nom du fichier externe est spécifié par le paramètre `name`.
- Le paramètre `form` fixe des propriétés pour le fichier (droits d'accès par exemple)

```
procedure create( file: in out File_type;
                  mode: in File_mode:= out_file;
                  name: in String:= "";
                  form: in String:= ""; );
```

Ouverture/fermeture d'un fichier texte

```
procedure open  ( file: in out File_type;
                  mode: in File_mode;
                  name: in String:= "";
                  form: in String:= ""; );

-- suppression de l'association
  fichier/fichier
-- externe
procedure close ( file:in out File_type );
```

Accès aux éléments d'un fichier texte

```
with Ada.Text_io; use Ada.Text_io;
procedure exemple is
   fichier:File_type;
   cat : Character;
begin
   open (fichier,in_file,"texte.dat" );
   while not end_of_file( fichier ) loop
        while not end_of_line( fichier ) loop
               get( fichier,car );
               put( car );
        end loop;
        skip_line;
   end loop;
   close( fichier );
end exemple;
```

get_immediate

- La procédure `get_immediate` permet de lire un caractère sans attendre que le caractère de fin de ligne soit tapé.
- Le programme attend tant qu'aucun caractère n'est disponible.

```
get_immediate(item:out Character);
```

- Dans cette version, le paramètre `available` est mis à `false` si aucun caractère n'est disponible et l'exécution se poursuit.

```
get_immediate(item:out Character;
              available:out Boolean);
```

Fichier binaires

- Ils sont constitués d'une suite de bits auxquels seuls des programmes adapté peuvent donner un sens.

- Le traitement de tels fichiers s'effectue grâce aux paquetages `Ada.sequential_io` pour un accès séquentiel et `Ada.direct_io` pour un accès direct.

- La manipulation de fichiers binaires nécessite la connaissance du type des éléments enregistrés.

- Modes d'accès : `in_file, out_file, append_file, inout_file` (permet de lire, modifier, ajouter des informations n'importe où dans le fichier)

Création/ouverture/fermeture de fichier binaire

```
procedure create( file: in out File_type;
                  mode: in File_mode:= ...;
-- par défaut out_file pour les fichiers séquentiels
-- inout_file pour les fichiers en accès direct
                  name: in String:= "";
                  form: in String:= ""; );
procedure open  ( file: in out File_type;
                  mode: in File_mode;
                  name: in String:= "";
                  form: in String:= ""; );

-- suppression de l'association fichier/fichier
-- externe
procedure close ( file:in out File_type );
```

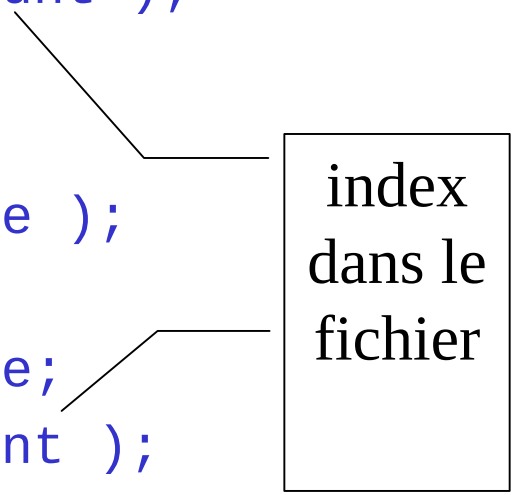
Accès séquentiel : lecture/écriture

```
procedure read( file: in File_type;
                item: out Element_type);

procedure write( file: in File_type;
                 item: in Element_type);
```

```
with Ada.sequential_io;use Ada.sequential_io;
with dates; use dates;
-- exporte le type Date et la procedure put
procedure exemple_sequentiel is
  fichier_dates:File_type;
  package dates_io is new Ada.sequential_io(Date);
  use dates_io;d:Date;
begin
  create( fichier_dates, "dates.dat" );
  write( fichier_dates,(13,5,1972) );
  write( fichier_dates,(2,11,1965) );
  close ( fichier_dates );
  open( fichier_dates,in_file, "dates.dat" );
  read(fichier_dates,d );
  put( d );
  read(fichier_dates,d );
  put( d );
  close (fichier_dates );
end exemple_sequentiel;
```

Accès direct : lecture/écriture

```
subtype Positive_count is Natural range 1..Natural'last;
procedure read( file: in File_type;
                item: out Element_type );
procedure read( file: in File_type;
                item: out Element_type;
                from : in Positive_count );

procedure write( file: in File_type;
                 item: in Element_type );
procedure write( file: in File_type;
                 item: in Element_type;
                 to : in Positive_count );
```



```
with Ada.direct_io;use Ada.direct_io;
with dates; use dates;
-- exporte le type Date et la procedure put
procedure copie_fichier is
  original,copie:File_type;
  package dates_io is new Ada.direct_io(Date);
  use dates_io;d:Date;
begin
  open( fichier_dates,in_file, "dates.dat" );
  create( fichier_dates, "copie.dat" );
  for i in reverse 1..size(original) loop
    set_index( fichier_dates,i);
    read( original,d );
    write( copie,d );
  end loop;
  close( original );
  close( copie );
end copie_fichier ;
```