**TE 9**

# Structures en fortran 2003 et C

## 1 Exercice élémentaire : conversion de dates A

Afin de prendre en main l'utilisation des structures et des chaînes de caractères, on se propose de créer un programme simple, qui permet de convertir un numéro de jour en date au format jour-mois-année. Le numéro du jour correspond au nombre de jours écoulés depuis le 1er janvier de l'an zéro en considérant pour simplifier qu'une année se compose de douze mois de trente jours. La date sous la forme jour-mois-année sera représentée par une structure ou type dérivé nommé `date` composé de trois champs [1] :

- `jour`, de type entier
- `mois`, de type chaîne de caractères comportant 4 caractères (sans accent) : `janv`, `fevr`, `mars`, `avri`, `mai␣`, `juin`, `juil`, `aout`, `sept`, `octo`, `nove`, `dece`
- `annee`, de type entier

## 2 Analyse de radiosondages A

On se propose de créer des procédures pour manipuler des fichiers de radiosondages atmosphériques [2]. L'ensemble des paramètres mesurés à **un niveau d'altitude** (voir entête du fichier) sera représenté par une structure ou type dérivé nommé `radios` comportant six champs dont le type et l'identifiant sont précisés ci-dessous :

- la pression en hPa (réel) `pression`
- l'altitude en m (entier) `alt`
- la température en Celsius (réel) `temp_c`
- la direction du vent en degrés (entier) `direct`
- la vitesse du vent en noeuds (entier) `vitesse`
- la température potentielle en Kelvin (réel) `theta`

Le fichier de radiosondage est un fichier texte dans lequel on a, pour simplifier, supprimé l'entête qui avait la forme suivante, où seul le premier niveau mesuré (à 168 m) est représenté.

```
-----------------------------------------
   PRES   HGHT   TEMP   DRCT   SKNT   THTA
    hPa      m      C    deg   knot      K
-----------------------------------------
  969.0    168    3.0    160      4  278.6
```

Un niveau est représenté par une ligne du fichier texte. Les données d'un sondage seront alors représentées par un tableau dynamique de structures qui comportera autant d'éléments que le fichier comporte de lignes.

1. En langage C, afficher la taille des champs d'un objet de type `date` ainsi que celle de la structure et expliquer.
2. Informations et données disponibles sur le site `http://weather.uwyo.edu/upperair/sounding.html`

**a)** Définir, dans un fichier d'entête en C, et dans un module en fortran, le type dérivé `radios` représentant les six paramètres mesurés à **un** niveau d'altitude.

**b)** Écrire une procédure `lect_sond` de lecture du fichier de radiosondage qui admet comme paramètres d'entrée le nom du fichier de radiosondage et le nombre de niveaux. Elle alloue le tableau de structures, lit le fichier et affecte ce tableau de façon à obtenir une image structurée du fichier en mémoire. Elle ferme alors le fichier lu. Il est prudent de lui faire afficher les valeurs lues.

**c)** Compter le nombre de lignes du fichier texte `trappes-04dec2011-00h.dat`. Appeler la procédure `lect_sond` dans le programme principal pour lire ce fichier et vérifier les valeurs affichées.

**d)** Rechercher l'altitude du minimum de température du sondage et l'afficher.

**e)** Créer une procédure `calc_theta` qui, à partir des données du radiosondage calcule la température potentielle $\theta$ via l'expression :

$$\theta = T \left[\frac{p_0}{p}\right]^{R/C_p} \quad \text{où} \quad R/C_p \approx 2/7$$

$T$ est la température absolue en Kelvin, $p$ la pression et $p_0$ la pression de référence (1000 hPa). On pourra comparer avec la température potentielle déjà disponible dans le radiosondage. Cette procédure écrira les profils verticaux de température absolue et de température potentielle dans un fichier nommé `temp.dat`, sous la forme : altitude, température, température potentielle avec un niveau par ligne.

**f)** Tracer ces deux profils verticaux $T(z)$ et $\theta(z)$ avec `gnuplot`.

**g)** B Créer une procédure `calc_vent` qui, à partir des données du radiosondage calcule les deux composantes horizontales $u$ et $v$ du vent. Elle écrira le profil vertical de ces composantes dans le fichier donné en argument. Tracer ces deux profils verticaux $u(z)$ et $v(z)$ avec `gnuplot`.

## Interfaces à respecter

```
subroutine lect_sondage(file_name, nb_niv, t_niv)
  character(len=*), intent(in) :: file_name
  integer, intent(in) :: nb_niv
  type(radios), dimension(:), allocatable, intent(out) :: t_niv
end subroutine lect_sondage

subroutine calc_theta(file_name, t_niv)
  character(len=*), intent(in) :: file_name
  type(radios), dimension(:), intent(in) :: t_niv
end subroutine calc_theta

subroutine calc_vent(file_name, t_niv)
  character(len=*), intent(in) :: file_name
  type(radios), dimension(:), intent(in) :: t_niv
end subroutine calc_vent
```

```
struct radios * lect_sondage( const char* file_name, int nb_niv);

void calc_theta(const char* file_name, int nb_niv, struct radios* t_niv);

void calc_vent(const char* file_name, int nb_niv, struct radios* t_niv);
```

## 3 Histogrammes B

L'histogramme de la distribution d'un échantillon d'une variable aléatoire est caractérisé par un ensemble d'intervalles (ou classes) et le nombre de valeurs (ou population) de l'échantillon situé dans chacun de ces intervalles. Dans le cas où la variable est à valeurs réelles, l'histogramme peut donc être représenté par une structure nommée `histo` constituée de deux tableaux :
– un tableau `bornes` de réels, les bornes finies des intervalles
– et un tableau `population` d'entiers, les populations de ces intervalles.
Afin de pouvoir traiter des variables à support non fini, on préservera systématiquement deux intervalles ouverts s'étendant l'un vers $-\infty$, l'autre jusqu'à $+\infty$. Pour une variable à support fini connu, on choisira les bornes de façon à ce que les classes extrêmes restent vides. En revanche, l'adjonction de ces classes permet de ne pas être contraint de calculer les valeurs extrêmes de la distribution avant de lancer un calcul d'histogramme : la première et la dernière classe permettront de stocker les valeurs en dehors de l'intervalle fini fixé a priori.
Compte tenu des différences d'indexation dans les deux langages, la structure d'histogramme `histo` prend donc la forme :
– En fortran,

$$\begin{array}{cccc} ]-\infty, \texttt{bornes(1)}[, & [\texttt{bornes(1)}, \texttt{bornes(2)}[, & \cdots, & [\texttt{bornes(N)}, +\infty[ \\ \texttt{population(1)} & \texttt{population(2)} & \cdots & \texttt{population(N+1)} \end{array}$$

– En langage C,

$$\begin{array}{cccc} ]-\infty, \texttt{bornes[0]}[, & [\texttt{bornes[0]}, \texttt{bornes[1]}[, & \cdots, & [\texttt{bornes[N-1]}, +\infty[ \\ \texttt{population[0]} & \texttt{population[1]} & \cdots & \texttt{population[N]} \end{array}$$

Les types dérivés seront définis de façon globale en C et encapsulés dans un module en fortran.
Les échantillons pseudo-aléatoires seront tirés comme dans l'exercice sur les moments :
– En fortran, par **un** appel au sous-programme intrinsèque `random_number` avec un argument de type tableau de réels. Il fournit des réels pseudo-aléatoires distribués uniformément dans l'intervalle [0,1[.
– En langage C, par une suite d'appels à la fonction `rand` qui rend un entier distribué uniformément dans l'intervalle [0, `RAND_MAX`], que l'on ramenèra dans l'intervalle [0,1[.
On ramenera chacun de ces tirages dans l'intervalle $[-1/2, +1/2]$. Pour varier les distributions, on pourra faire subir à ces échantillons des transformations telles que l'élévation à une puissance entière avant de calculer leur histogramme.
Créer et tester les procédures :

**a)** `creer_histo_regulier` de construction de l'histogramme à pas constant d'un tableau pseudo-aléatoire `x` de dimension variable `nb_tirages`. La procédure recevra en entrée les bornes finies `xmin` et `xmax` d'analyse et construira la structure `histo_x` de type `histo`

**b)** `afficher_histo` d'affichage d'un histogramme quelconque avec une ligne par classe indiquant les bornes de classe et la population pour chaque intervalle.

**c)** `ecrire_histo` d'écriture de la partie de l'histogramme ne comportant pas les deux intervalles extrêmes dans un fichier `histo.dat`. L'histogramme pourra alors être tracé avec `gnuplot` en relisant ce fichier.

### 3.1 Nombre de classes fixe

La taille des tableaux `bornes` et `pop` est fixée à la compilation :
– En fortran via une constante nommée, définie dans le module `m_histogramme`
– En langage C grâce à un `#define` traité par le préprocesseur ;

### 3.2 Nombre de classes variable

Le nombre de classes est alors déterminé lors de l'exécution du programme, donc les tableaux des limites de classes et de leurs populations doivent être alloués lors de l'exécution. En particulier, la procédure `creer_histo_regulier` devra prendre en charge ces allocations.
– En fortran, on travaillera dans la norme 2003, qui autorise les tableaux allouables [3] comme composantes de types dérivés.
– En langage C, on pourra s'appuyer sur les fonctions `float1d`, `int1d` et `float1d_libere` et `int1d_libere` de la bibliothèque `libmnitab` pour gérer les tableaux dynamiques de rang 1.

### 3.3 (⋆) Recopies de structures

Les comportements des langages C et fortran (standard 2003) diffèrent profondément dans le cas des structures dynamiques. Fortran 2003 se charge si nécessaire de l'allocation « au vol » des composantes des structures en cas d'affectation vers une structure non allouée ou allouée avec un profil différent de celui du membre de droite. Au contraire, comme la structure C ne comporte qu'un pointeur vers les données, la copie reste superficielle : on ne fait que dupliquer le pointeur sans dupliquer les données. Il faut donc prendre en charge la copie profonde.
– Tester la recopie globale d'une structure d'histogramme par l'opérateur d'affectation (`=`) dans le cas de structures de taille fixe et dans le cas dynamique. Comparer les histogrammes obtenus via les fichiers créés par `ecrire_histo`.
– Pour le cas des structures dynamiques en C, écrire une procédure de libération de la mémoire allouée par les structures de type `histo` : cette procédure mettra à `0` le nombre de bornes et à `NULL` les champs de type pointeurs. Dans le cas dynamique, libérer la copie et écrire de nouveau l'histogramme original : conclure. Remplacer l'affectation (=) par une fonction de copie réelle (profonde) des données de la structure d'histogramme et refaire le test de libération de la copie.

3. En fortran 95, il faudrait soit utiliser une option comme `-ftr15581` avec `g95` par exemple ou faire appel à des pointeurs.