# Les types utilisateurs (F)
# Corrigé

**Résumé**

Ce document décrit comment traduire en F les types utilisateur du langage algorithmique.

## Table des matières

# Liste des exercices

# 1 Types énumérés

Les types énumérés n'existent pas en F. On peut donc utiliser les entiers et définir une constante entière par valeur du type énuméré.

Suivant le contexte, on pourra prendre un autre type, par exemple caractère.

# 2 Les tableaux

## 2.1 Motivation

## 2.2 Définition

## 2.3 Tableaux à une dimension

Les tableaux en F sont beaucoup plus contraignants qu'en algorithme car le type des indices est forcément entier. Pour déclarer un tableau en F, il suffit de donner le type des éléments et la capacité.

```
1 REAL,DIMENSION(10)::vect        ! vect est un tableau de 10 réels
```

Les indices valides sont donc 1, 2, ..., 9 et 10. C'est équivalent à la déclaration suivante en algorithme :

```
1 Variable
2     vect: Tableau [1..10] De Réel
```

Comme en algorithmique, il est possible de définir les deux bornes de l'indice du tableau :

```
1 REAL,DIMENSION(0:9)::vect ! vect est un tableau de 10 réels
```

**Remarque :** Il est fortement conseillé d'utiliser une constante symbolique à la place de la constante littérale 10 ou 9.

**Accès à un élément :** L'accès à un élément se fait en utilisant des parenthèses :

```
1 vect(5)=10.0
2 vect(1)=vect(5)+1
```

Il existe des notions particulières en F : `vect(1:3)` représente une section du tableau `vect` (ce sont les trois premiers éléments du tableau `vect`. L'affectation entre tableaux de même dimension est possible

```
1 vect(1:3)=vect(5:7)
```

par extension on peut initialiser un tableau à 0 par `vect=0` pour être plus clair il est fortement conseillé de l'écrire `vect(:)=0` ce qui indique que c'est un tableau).

F étant un langage de calcul scientifique de nombreuses opérations sur les tableaux sont possibles.

**Attention :** Aucun contrôle n'est fait sur la validité des indices utilisés. Des indices incorrects correspondent à un programme faux. Malheureusement, ceci ne provoque pas nécessairement un plantage du programme. Ceci rend ces erreurs difficiles à identifier. Il faut donc être vigilant : à chaque fois que vous accéder à un élément d'un tableau, demandez-vous si l'indice est valide.

Notez que pour être plus strict, il est possible d'indiquer au compilateur de vérifier que les indices sont dans les bornes de la définition en utilisant les options `-C` et `-gline`.

## 2.4 Exemples

**Exercice 1 : Initialiser un tableau**

Initialiser astucieusement un tableau de 10 entiers pour qu'il contienne les valeurs suivantes :

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|

**Solution :** Si on considère un tableau définit par :

```
Type
    Vecteur = Tableau [1..10] De Entier
```

On constate que la valeur de la case d'indice $i$ est justement $i$. Pour initialiser un vecteur, il suffit donc de parcourir les entiers de 1 à 10 pour remplir la case correspondante.

```
Variable
    tab: Vecteur
    indice: Entier
Début
    Pour indice <- 1 JusquÀ indice = 10 Faire
        tab[indice] <- indice
    FinPour
Fin
```

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.1
!* Revision : Xuan Meyer <XuanMi.Meyer@ensiacet.fr>
!*
!* Objectif : Initialiser un tableau avec dans l'ordre les valeurs :
!* 1, 2, 3, 4, 5, 6, 7, 8, 9 et 10.
!*********************************************************************

PROGRAM main

    INTEGER,DIMENSION(10)::tab   ! le tableau à initialiser
    INTEGER:: indice   ! parcourir les entiers de 1 à 10
    DO indice=1,10   ! le pas est omis quand il est égal à un
        tab(indice)=indice
    ENDDO
    END PROGRAM main
```

**Exercice 2 : Afficher un tableau d'entier**

Écrire un programme qui affiche tous les éléments d'un tableau de capacité MAX (égale à 10) mais dont la taille effective (c'est-à-dire le nombre d'éléments utiles) est donné par la variable `nb`.

Les éléments du tableau seront écrits entre crochets, dans l'ordre croissant de leur indice et séparés par des points-virgules. Voici quelques exemples :

```
[]              -- un tableau vide (de taille effective nulle)
[ 1; 2; 3]      -- tableau contenant les 3 valeurs 1, 2 et 3.
[ 421 ]         -- tableau contenant la seule valeur 421
```

**Solution :** On constate que l'on ne peut pas afficher directement chaque élément du tableau car le point-virgule ne doit pas être mis après le dernier élément. le principe est donc d'afficher d'abord le premier élément puis, pour tous les éléments suivants, d'afficher le point-virgule séparateur et l'élément lui-même. Ceci ne peut bien sûr être fait que si le tableau contient au moins un élément.

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.2
!* Revision : Xuan Meyer <XuanMi.Meyer@ensiacet.fr>
!* Objectif : afficher un tableau d'entiers de taille effective nb
!*********************************************************************

PROGRAM main
    INTEGER,PARAMETER::max=10
    INTEGER:: nb=3  ! nous aurions pu utiliser une constante dans ce cas
    INTEGER,DIMENSION(max)::tab  ! le tableau à afficher
    ! remarque utilisant d'un constructeur de tableau pour l'initialiser
    INTEGER:: i   ! parcourir les entiers de 1 à nb

    ! utilisation d'un constructeur de tableau pour l'initialiser (//) et
    ! d'une section de tableau car la dimension doit être identique
    tab(1:nb) = (/4,2,1/)

    ! Afficher le tableau tab de taille nb
    PRINT*,"[",(tab(i),";",i=1,nb-1),tab(nb),"]"
    ! utilisation d'une boucle implicite pour afficher le tableau sur la même
    ! ligne

END PROGRAM main
```

## 2.5 Chaînes de caractères

Les chaînes de caractères sont définies comme les caractères vus précédemment en modifiant la longueur (**LEN=**). Elles sont donc de longueurs définies et complétées par des espaces à droite **LEN**(ch) représente la longueur de la chaîne ch (la même que dans la définition) TRIM(ch) représente la chaîne débarrassée de ces espaces à droite.

LEN_TRIM(ch) représente la longueur de la chaîne débarrassée de ces espaces à droite. L'opérateur de concaténation se note //

Les opérateurs de comparaison sont utilisables (code ASCII)

Une sous chaîne est définie comme une section de tableau ch(3:5) désigne la sous-chaîne comportant les 3, 4 et 5$^{e}$ caractères de la chaîne ch. ch(4:4) représente le quatrième caractère de la chaîne ch.

## 2.6 Tableaux à plusieurs dimensions

Un tableau à plusieurs dimensions se déclare comme en mathématiques :

```
REAL,DIMENSION(10,20)::m1,m2,m3
        ! par convention mathématique on suppose 10 lignes et 20 colonnes
! Pour faire m3=m1+m2
```

# 3 Les enregistrements

Le principe est strictement le même qu'en algorithmique. C'est le signe % qui permet d'accéder aux champs

```
1  !**********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.2
4  !* Revision : Xuan Meyer <XuanMi.Meyer@ensiacet.fr>
5  !*
6  !* Objectif : Type derives
7  !**********************************************************************
8  PROGRAM main
9
10     TYPE:: Date
11         INTEGER:: jour ! numero du jour dans le mois
12         INTEGER:: mois ! numero du mois dans l'annee
13         INTEGER:: annee ! numero de l'annee
14     END TYPE date
15
16     TYPE(date):: d1, d2 ! deux dates
17     INTEGER::annee
18
19     ! initialiser la date d1
20     d1%jour = 23
21     d1%mois = 11
22     d1%annee = 2000
23     ! initialiser la date d2 à partir de d1
24     d2 = d1
25     ! afficher la date d2
26     PRINT*, d2%jour,"/", d2%mois,"/", d2%annee
27     ! conserver l'annee de d2
28     annee = d2%annee
29     ! saisir l'annee de d2
30     PRINT*, "saisir␣l'annee␣de␣d2"
31     READ*,d2%annee
32     ! afficher la date d2
33     PRINT*, d2%jour,"/", d2%mois,"/", d2%annee
34
35 END PROGRAM main
```