1 Variables statique ou dynamiques

Variable statique : memoire reservee a la compilation
Variable dynamique : memoire reservee a l'execution

Variables et initialisations: data et save
Declarer les variables par type et centre dinteret suivant leur fonction en haut du code

- declaration **dynamique**
- declaration et affectation **statique**
- save pour garder la valeur des variables entre chaque passage dans une subroutine.

Exemple: realisation dun compteur interne

```
subroutine sub(..)
integer :: icom=0
real,save :: pi
if (icom .eq. 0) then
pi = 4.*atan(1.)
endif
...
icom = icom + 1
return
end
```

Le save icom nest pas necessaire car icom etant initialisee devient statique. Par contre le save pi lui est necessaire.

2 Unités de programme

Principal

```
1234567
      program mm
      implicit none
      integer n,i
      real y
      double precision x
     & ,z
        ...
```

sous-programme

```
1234567
      subroutine trap(n,a,b,f,s)
      implicit none
      integer n,i
      real y
       ...
      end

      call trap(n,a,b,f,s)
```

fonction

```
1234567
      real function sinx(x)
      implicit none
      real x

      sinx = sin(x)/x
      return
      end
```

Aide-memoire Fortran tableaux

Jean-Guy Caputo

3 Tableaux

dynamique

```
real, dimension (:), allocatable :: a
```

pour allouer

```
allocate (a(n))
```

La **méthode de travail** est

1. on lit la dimension
2. on alloue le tableau
3. on remplit le tableau
4. on travaille
5. on desalloue deallocate (a)

Regle de travail
On alloue dans le main, pas dans les sous-programmes Les tableaux de travail sont passes une fois alloues dans le main. Ceci rend les sous-programmes portables, indépendants de la dimension.

Tableau statique

```
integer, parameter :: n=5
real a(n)
```

Ordre de stockage des tableaux en mémoire a(n,n)
$a_{11}a_{21}a_{31}\ldots$
l'indice le plus a gauche varie d'abord, en C c'est l'inverse

4 Operations sur des tableaux conformants

Profil d'un tableau
shape (SOURCE) is a function which returns the shape of an array SOURCE as an integer vector.

Tableaux conformants si meme profil

Un scalaire est conformant a tout tableau

```
...
integer, parameter :: n=5
real a(n),b(n),c(n)

...
a = 0.
a = b*c+d

r = dot_product(a,b)
```

operations importantes
matmul , dot_product, transpose

En MATLAB

$A. * B \neq A * B$

dot_product $u * v'$

transposee d'une matrice A A'

5 Instruction allocate : tableaux dynamiques

allocate (array, stat=err) permet de faire l'allocation mémoire du tableau. Si le mot clé stat= est spécifié, la variable *err* (de type integer) vaut 0 si l'allocation s'est déroulée sans problème et 1 sinon.

deallocate (array, stat=err) permet de libérer l'espace mémoire réservée à un tableau.

allocated(array) est une fonction intrinsèque renvoyant true ou false suivant que le tableau spécifié en argument est alloué ou non.

```
integer, dimension (:,:), allocatable :: a
integer :: n, m, err

read (*,*)  n, m
if (.not. allocated(a)) then
   allocate(a(n,m),stat=err)
   if (err /= 0) then
     write(*,*) 'Erreur d''allocation dynamique du tableau a'
     stop
   endif
endif
...
dallocate(a)
```

6 Fonctions information sur des tableaux

lbound (A,dim) is a function which returns the lower dimension limit for the array A. If dim (the dimension) is not given as an argument, you get an integer vector, if dim is included, you get the integer value with exactly that lower dimension limit, for which you asked.

shape (SOURCE) is a function which returns the shape of an array SOURCE as an integer vector.

size (ARRAY, dim) is a function which returns the number of elements in an array ARRAY , if DIM is not given, and the number of elements in the relevant dimension if DIM is included.

lbound (array [,dim]) et **ubound**(array [,dim]) retournent les bornes inférieures / supérieures de chacune des dimensions (ou seulement de la dimension *dim*) du tableau *array*.
Exemple :
integer, dimension(21:2, 45:49) :: tab
lbound(tab) vaut (/ 21, 45 /) et ubound(tab) vaut (/ 2, 49 /)
lbound(tab, dim=2) vaut 45 et ubound(tab, dim=1) vaut 2

all ALL(mask [,dim]) applique un masque de type logique sur les éléments du tableau et renvoit vrai si pour tous les éléments le résultat du masque est vrai. Si *dim* est précisé, la fonction travaille sur cet indice pour chaque valeur des autres dimensions.

any (mask [,dim]) fonctionne de la même façon que la fonction ALL à l'exception qu'elle renvoit vrai si l'un des résultats du masque est vrai.

count (mask [,dim]) comptabilise le nombre d'éléments pour lesquels le résultat du masque est vrai.

```
program test_all
logical l
integer a(2,3), b(2,3)

l = all((/.false., .true., .true./))
write(*,*)  l
l = any((/.false., .true., .true./))
write(*,*)  l

a = 1
b = 1
b(2,2) = 2
b(2,3) = 2
write(*,*)  all(a==b, 1)
write(*,*)  all(a .eq. b, 2)

write(*,*)  any(a .eq. b, 1)
write(*,*)  any(a .eq. b, 2)

write(*,*)  count(a /= b)
end
```

minval (array [,dim][,mask]) et **maxval**(array [,dim][,mask]) retournent la plus petite / plus grande valeur du tableau *array* (après un éventuel filtrage par *dim* et / ou *mask*).
exemple
minval(A,dim=2,A > 2) vaut (/ 3, 4 /)

minloc (array [,dim][,mask]) et **maxloc** retournent l'indice de la plus grande ou dela plus petite valeur.
Attention : maxloc dans des sections de matrice, il faut mettre dim.
exemple recherche du pivot dans la colonne k sous diagonale
ipiv = maxloc(D(k:n,k),1)+k-1

7 Initialisation de tableaux

```
integer,parameter::n=4
integer,dimension(n)::t1,t2,t3
```

```
integer i
t1=(/6,5,10,1/)
t2=(/ (i+1,i=1,n) /)
t3=(/ t2(1) t1(3) ,1,9  /)
```

8 Sections de tableaux

9 Operations sur des tableaux

product (array [,dim][,mask]) et **sum** (array [,dim][,mask]) retournent le produit / la somme des valeurs des éléments du tableau *array* (après un éventuel filtrage par *dim* et / ou *mask*).
Exemple :
product(A) retourne 720
sum(A,dim=1,A > 2) vaut (/ 4, 5, 9 /)

transpose (matrix), ou *matrix* est un tableau de dimension 2, retourne la transposée de matrix.

dot_product (a, b) retourne le produit scalaire de deux vecteurs, c'est à dire at.b si a et b sont de type entier ou réel, conjugué(a)t.b si a et b sont complexes, et Somme(ai .and. bi) si a et b sont de type logique.

matmul (a, b) retourne le produit matriciel de a et b, sous réserve de compatibilité des dimensions. Soit a et b sont tous deux des matrices, soit a est un vecteur et b est une matrice, soit a est une matrice et b est un vecteur. a et b peuvent être de type quelconque.

norm2 (a) retourne la norme euclidienne du tableau a (vecteur ou matrice).

10 Action sur des tableaux

L'instruction **where** permet d'effectuer des opérations sur des éléments d'un tableau sélectionnés via un filtre de type logique.

```
real, dimension(10) :: a

where (a > 0.) a = sqrt(a)
```

```
where (a > 0.)
  a = log(a)
elsewhere
  a = 1.
end where
```

Pour plus d'informations
https://gcc.gnu.org/onlinedocs/gfortran/Intrinsic-Procedures.html#Intrinsic-Procedures