Fortran 90 et 95

Dominique COLOMBANI - Web Formation

Table des matières

3. Variables

3.1. Présentation des variables

Variable Az Définition

- zone mémoire, identifiée par un nom
  - au maximum 31 caractères : lettres, chiffres et _
  - le premier caractère doit être une lettre
  - pas de différences entre majuscule et minuscule
- une variable a un type, défini une fois pour toute
  - spécifie ce qui peut être représenté (entier, caractère, réel, ...)
  - la façon dont est faite la représentation
  - les bornes des valeurs qui peuvent être représentées

Déclaration du type

- la déclaration doit être effectuée au début de l'unité de compilation
- la déclaration explicite n'est pas obligatoire, mais est fortement recommandé, pour éviter le typage implicite
- formes de la déclaration
  - type identifiant1, identifiant2 ! Fortran 77
  - type [,attribut1, atribut2] :: identifiant1, identifiant2 ! Fortran 90, avec des attributs optionnels

3.2. Typage implicite

- Les variables non déclarées ont un type implicite.
- Par défaut, la première lettre de l'identificateur définit le type

  I, J, K, L, M, N : entier (I-N, début de INTEGER)

  autre : réel

  Très dangereux ...
- Instruction IMPLICIT permet de définir d'autres typages par défaut

  IMPLICIT DOUBLE PRECISION (A-D)

  Très dangereux également

2. https://onlinegdb.com/C3ZILWY-S

Bonne pratique Conseil

- utiliser IMPLICIT NONE (ou l'option équivalente du compilateur) pour imposer la déclaration de chaque variable
- Options du compilateur

  gfortran : -fimplicit-none

  Intel : /warn:declarations (Windows) ou -warn declarations (Linux)

3.3. Types de base

3.3.1. INTEGER

- entier dont la taille dépend du compilateur
- constantes
  - décimale : 23
  - binaire : B'1101001111' (avec DATA)
  - octale : O'735' (avec DATA)
  - hexadécimale : Z'FA3' ou X'FA3' (avec DATA)
- spécification de taille

  Fortran 77 : INTEGER*2, INTEGER*4
- Fortran 90 : INTEGER(KIND=2, INTEGER(KIND=4) ou INTEGER(2), INTEGER(4)

3.3.2. REAL

- valeurs numériques sur 4 octets
- 1.2*10-38<= |r| <= 3.4*10+38 avec 6 chiffres significatifs.
- constantes :

  0.123

  .123

  1.23E-1

  -0.123
- spécification de taille

  Fortran 77 : REAL*4 (identique à REAL), REAL*8 (DOUBLE PRECISION)

  Fortran 90 : REAL(KIND=4) , REAL(KIND=8) ou REAL(4), REAL(8)

3.3.3. KIND

Détermination d'une valeur de KIND

- SELECTED_INT_KIND(m) : valeur de KIND pur les nombres compris entre -10^m et 10^m

  renvoie -1 si impossible
- SELECTED_REAL_KIND((p,r) : valeur de KIND pur les nombres tels que $10^{-r} < |x| < 10^r$ avec une précision p

renvoie

-1 si la précision demandée n'est pas disponible

-2 si l'étendue n'est pas disponible ;

-3 si la précision et l'étendue ne sont pas disponibles.

```
program select_kind
implicit none

print*,selected_int_kind(8)     ! 4
print*,selected_int_kind(2)       ! 1
print*,selected_int_kind(20)      ! 16
print*,selected_int_kind(40)      ! -1
print*,'Flottant ---------'
print*,selected_real_kind(8,5)    ! 8
print*,selected_real_kind(4,5)    ! 4
print*,selected_real_kind(4,100)  ! 8
print*,selected_real_kind(8,400)  ! 10
print*,selected_real_kind(8,5000) ! -2

end program select_kind
```

Exécuter ce code[3]

valeur de kind Remarque

- la norme n'impose pas que les valeurs de kind soient la taille en octets (4 ou 8). Il peut exister des compilateurs avec des valeurs 1, 2, ...

  (=> documentation du compilateur)
- on peut utiliser un paramètre pour kind

  integer, parameter :: r=4

  real(kind=r) :: x
- on peut définir une constante avec le type associé à kind avec la notation _kind

  x = 3 .0_r ; y=5.0_4

```
program demo_kind
implicit none
integer, parameter:: r=4
real(kind=r) :: x,y
x = 3.0_r ; y=5.0_4
print*,x,y

end program demo_kind
```

Exécuter ce code[4]

RANGE, PRECISION

RANGE fournit la plage de valeur

PRECISION fournit le nombre de chiffres significatifs

3. https://onlinegdb.com/d0ZGxV76qw

```
program demo_range_precision
implicit none
real :: r
real(8) :: d
integer :: i
integer(2) :: j
print*, range(r), precision(r)
print*, range(d), precision(d)
print*, range(i)
print*, range(j)
end program demo_range_precision
```

Exécuter ce code[5]

3.3.4. DOUBLE PRECISION

- valeurs numériques sur 8 octets
- 2.2*10-308 <=|r| <= 1.8*10+308 avec 15 chiffre s significatifs
- constantes :

  0.123D0

  .123D0

  1.23D-1

  -0.123D0
- peut être déclaré avec une instruction REAL*8 au lieu de DOUBLE PRECISION

```
program declar
implicit none
real*8 pi
pi = 3.14d0
write (6,*) pi
end program declar
```

Exécuter ce code

```
program precision
real pi /3.14/
real*8 double_pi /3.14D0/
real(kind=4) :: r
real(kind=8) :: r2
r = 50.
r2 = 5.D1
write(6,*) r * pi -157.
write(6,*) r2 * double_pi -157.D0
end program precision
```

Exécuter ce code

3.3.5. COMPLEX

- nombre complexe dont les parties réelle et imaginaire sont des REAL
- COMPLEX*16 pour des double précision
- constante :

  (2.5, 1.) représente 2.5 + i
- fonctions intrinsèques REAL et IMAG pour accéder aux parties réelles et imaginaires

---

4. https://onlinegdb.com/8h_cah0JM


```
Program demo_complexe
implicit none
complex :: c
c = (10.,1.)
print*,c ! (10.0000000,1.00000000)
print*, "Partie réelle ", real(c) ! 10.0000000
c = c * c
print*, c ! (99.0000000,20.0000000)
end
```

Exécuter ce code[6]

3.3.6. LOGICAL

- valeurs booléennes
- .TRUE. , .FALSE.

```
program prog
integer*4 i,j
logical ok, ko
ok = .true.
ko = .false.
write (6,*) ok, ko
i=1
j=2
write(6,'(1X,L10)') i .eq. j
end
```

Exécuter ce code

3.3.7. CHARACTER

- code ASCII
- définition d'une longueur
  - CHARACTER(LEN=n) nom
  - CHARACTER*n nom
  - CHARCTER nom*n
- constantes " " ou ' '
  (attention : chaîne complétée par des espaces lors d'une affectation)
- ancienne forme de constante, utilisée pour des initialisations ou des formats
  constantes Hollerith : DATA TXT /7Hbonjour/
  (nombre de caractères, lettre H et caractères)
- sous chaîne CH(5:7)
  si on ne spécifie pas la borne, la valeur minimale ou maximale est utilisée
  - CH(:3) est équivalent à CH(1 3)
  - CH(3:) est équivalent à CH(3:LEN(CH))

```
program demo_caractere
character(len=10) texte1
character*5 texte2

texte1="Bonjour"
texte2='Au revoir'

```

5. https://onlinegdb.com/1BZMDBkm-

```
print*,texte1         ! Bonjour
print*,texte2         ! Au re
print*,texte1(1:3)    ! Bon
print*,texte1//texte2 ! Bonjour   Au re

end program demo_caractere
```

Exécuter ce code[7]

character(*) Remarque

On peut utiliser character(*) dans plusieurs constextes

- la chaine est une constante

  characte(*), parameter :: fort = "Fortran"
- la chaine est un argument d'une procédure

  function f1(text)

  character(*) text

```
program demo_caractere
character(*), parameter :: texte1 = "Bonjour"
character :: texte2*5 ='Au revoir'
character(80) upper

print*,texte1         ! Bonjour
print*, len(texte1)   ! 7
print*,texte2         ! Au re
print*,upper(texte1)

end program demo_caractere

character(80) function upper(t)
character(*),intent(in) :: t
integer :: i
character(len(t)) :: tm
do i = 1, len(t)
    if (t(i:i) >= 'a' .and. t(i:i) <= 'z') then
        tm(i:i) = achar(iachar(t(i:i))-32)
    else
        tm(i:i) =t(i:i)
    end if
end do
upper = tm
end function upper
```

Exécuter ce code

3.4. Initialisation

Fortran 77

- On peut initialiser une variable avec l'instruction DATA

  DATA liste_variables / liste_valeurs /
- les valeurs doivent correspondre aux types des variables

6. https://onlinegdb.com/-PqlxQkpST

- Une variable initialisée a souvent une durée de vie égale à celle du programme (attribut SAVE automatique) : cela dépend du compilateur
- L'initialisation peut être faite lors de la déclaration

  REAL PI /3.14/

Fortran 90

- Initialisation REAL :: PI = 3.14 ! les :: sont obligatoires

3.5. Portée et durée de vie

Portée

- Une variable est définie dans une unité de compilation
- Elle n'est pas connue en dehors de cette unité

```
1  program portee
2  implicit none
3  real :: x
4  x = 3.
5  call ssprog1
6  end program portee
7
8  subroutine ssprog1
9  implicit none
10 print*,x  ! Error: Symbol ‘x’ at (1) has no IMPLICIT type
11 end subroutine ssprog1
```

Exécuter ce code[8]

Durée de vie

- Par défaut, une variable (donc la valeur qu'elle a) est détruite à la fin de l'unité où elle est déclarée.
- Pour la rendre persistante, il faut utiliser SAVE
- Fortran 77

  REAL X

  SAVE X
- Fortran 90

  REAL, SAVE :: X
- attention

  aux initialisations qui font un SAVE implicite

  aux options de complication (-fautomatic, -fno_automatic de gfortran)

```
1  program duree_de_vie
2  call affiche
3  call autre
4  call affiche
5  end
6  subroutine affiche
7  real :: x = 3.
8  real :: y
9  print*, 'Valeur de x dans affiche : ',x
10 print*,  'Valeur de y dans affiche : ',y
11 x = 10
```

7. https://onlinegdb.com/sqkPbJLvF

```
y = 42
end
subroutine autre
integer :: i
i = 1000
print*, 'Valeur de i dans autre : ',i
end subroutine autre
```

Exécuter ce code[9]

NB : tester en commentant l'appel à autre

3.6. Constantes

Fortran 77

- On peut définir des constantes avec l'attribut PARAMETER
- REAL PI

  PARAMETER (PI=3.14)
- on peut utiliser des expressions, par exemple 22/7 ou 4*ATAN(1.)

Fortran 90

REAL, PARAMETER :: PI = 3.14

3.7. Synonymes

- On peut définir des synonymes avec l'instruction EQUIVALENCE
- Il y a juste correspondance des emplacements de variables, sans conversion
- COMPLEX C

  REAL R

  EQUIVALENCE (R,C)

```
program equiv
complex :: c = (3.,2.)
real :: r
equivalence (r,c)
print *,r
print *,c
r = 0
print *,c
end
```

Exécuter ce code

8. https://onlinegdb.com/TZfi3mMSh