

# Programmation Fortran

Serra Sylvain – 2024-2025



**ENSGTI**  
ÉCOLE D'INGÉNIEURS

Fortran

Serra Sylvain

Généralités

Remarques  
Logigramme  
Structure  
d'un  
programme  
Syntaxe  
générale  
Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- 1 Généralités
  - Quelques commentaires préliminaires et un peu d'historique
  - Logigramme
  - Structure d'un programme
  - Syntaxe générale
  - Compilation

2 Types de données

3 Opérateurs

4 Algorithmique

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

8 Suppléments

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ L'ordinateur ne comprend pas grand chose seul (rien en fait) et ne fait que ce que vous lui demandez.



Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

▶ En particulier, il :

- ne détecte pas les incohérences "logiques" en dehors des fautes de programmation ;  
 ex : utilisation d'une méthode ou d'une variable inconnue, demande d'ouverture et/ou d'écriture dans un fichier dont le nom n'est pas donné, écrasement de la valeur d'une variable. . .
- ne peut repérer à priori les incohérences mathématiques ou physiques d'une commande ;  
 ex : division par zéro, racine carrée négative . . .

▶ Enfin, il se doit d'obéir aux règles de fonctionnement de base du système d'exploitation (virus informatiques exceptés évidemment) et suit donc la logique intrinsèque de celui-ci.

⇒ *petit voyage spatio-temporel pour découvrir le fonctionnement d'un ordinateur. . .*

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

[1954 :] L'équipe de John Backus, d'IBM, publie le langage FORTRAN (FORMula TRANslator).

[1956 :] Premier manuel définissant le FORTRAN I.

[1957 :] Apparition du FORTRAN II et des premiers compilateurs commerciaux.

[1958 :] IBM développe, mais ne divulgue pas, FORTRAN III.

[1962 :] Sortie de FORTRAN IV qui se répand parmi les utilisateurs.

Développement de nouveaux compilateurs par d'autres constructeurs, manque de norme...

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

[1966 :] Création de la norme ANSI (American National Standards Institutes), qui rebaptise le FORTRAN IV en FORTRAN 66.

[1978 :] Fin des cartes perforées et modernisation du langage sous la forme du FORTRAN V ou FORTRAN 77.

[1991 :] Profonde modification avec la norme Fortran 90.

[1995 :] Apparition du Fortran 95.

[2004 :] Apparition du Fortran 03.

[2010 :] Apparition du Fortran 08.

[2018 :] Apparition du Fortran 18.

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Un logigramme est une représentation d'un processus en séquences distinctes, composées d'éléments spécifiques pré-définis.

*n.b.* pas réservé à l'informatique puisqu'on le trouve en qualité, suivi client, protocolisation (production, soins, prise en charge, etc.)

Commençons par travailler sur papier!



Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale  
Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

## En résumé, ...

- ▶ c'est un "pseudo-langage" entre notre langue (français, anglais, etc.) et les langages informatiques (Fortran, C, C++, Python. . . ) ;
- ▶ il permet d'exprimer clairement les étapes et actions à réaliser par un logiciel ;
- ▶ il peut se présenter sous forme d'un ordinogramme.

- Fortran
- Serra Sylvain
- Généralités
- Remarques
- Logigramme**
- Structure d'un programme
- Syntaxe générale
- Compilation
- Types de données
- Opérateurs
- Algorithmique
- Sous-programmes
- Tableaux
- Entrées – Sorties
- Suppléments
- Informations utiles

► Les éléments essentiels sont :



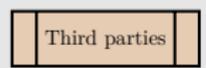
Début / Fin de processus



Action / Tâche à réaliser



Test / Condition de type Oui ou Non



Sous-programme / Processus extérieur



Entrées / Sorties



Suite

► Il en existe de nombreux autres (ignorés pour le moment)

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

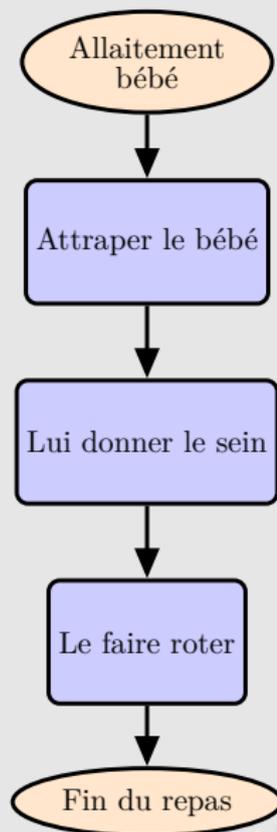
Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles



# Exemple (plus) complet

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure d'un programme

Syntaxe générale

Compilation

Types de données

Opérateurs

Algorithmique

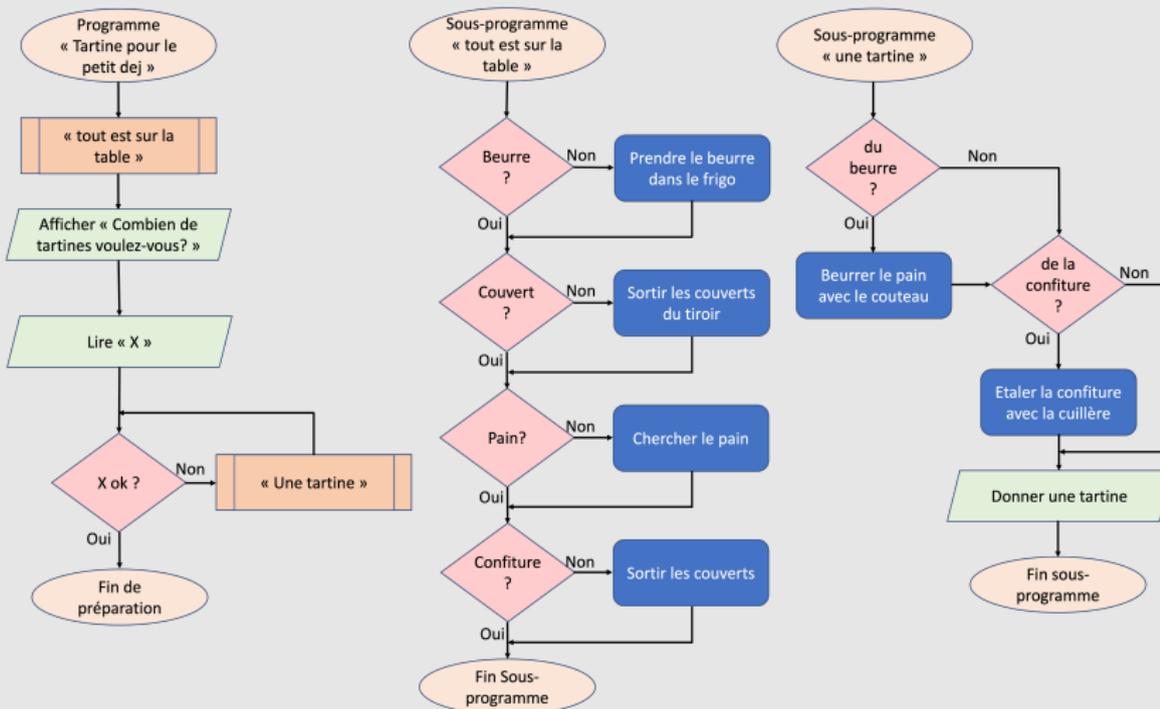
Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles



Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ **TOUJOURS** préparer un logigramme (même sommaire) **AVANT** de coder
- ▶ Le tester rapidement
- ▶ *Balayer succinctement* chaque étape pour discuter de sa *réalisation concrète*  
*n.b.* si vous doutez sur une étape, il est peu probable que la solution vienne en tapotant sur le clavier ou en regardant l'écran. . .

Fortran

Serra Sylvain

Généralités  
 Remarques  
 Logigramme

Structure  
 d'un  
 programme

Syntaxe  
 générale  
 Compilation

Types de  
 données

Opérateurs

Algorithmique

Sous-  
 programmes

Tableaux

Entrées –  
 Sorties

Suppléments

Informations  
 utiles

- ▶ Un programme est un ensemble d'instructions commandant la réalisation d'une série d'actions, éventuellement dépendantes de choix de l'utilisateur, définies **initialement et définitivement** par le programmeur.
- ▶ Le fortran est un langage séquentiel : le déroulement du programme suit l'ordre chronologique des instructions codées par le programmeur.

## Syntaxe

```
PROGRAM Nom  

Ensemble des déclarations  

Corps du programme  

END PROGRAM Nom
```

## Exemple

```
PROGRAM Elementaire  

END PROGRAM Elementaire
```

Fortran

Serra Sylvain

Généralités  
 Remarques  
 Logigramme

Structure  
 d'un  
 programme

Syntaxe  
 générale  
 Compilation

Types de  
 données

Opérateurs

Algorithmique

Sous-  
 programmes

Tableaux

Entrées –  
 Sorties

Suppléments

Informations  
 utiles

⚠ Les déclarations se font **toujours** au début du programme.

- ▶ Tout programme se décompose en fait en deux unités de programme :
  - ① le programme principal (**unique** et **obligatoire**).
  - ② les sous-programmes (**multiples** et **facultatifs**)
    - ① procédures (SUBROUTINE)
    - ② fonctions (FUNCTION)
    - ③ modules (MODULE)
    - ④ interfaces (INTERFACE)
- ▶ Les sous-programmes obéissent aux mêmes règles syntaxiques que le programme principal. On a une partie déclarative suivie d'une partie exécutive.

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

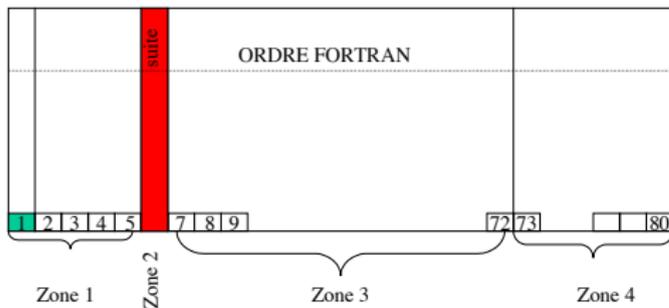
Suppléments

Informations  
utiles

- ▶ Il existe deux formats différents dans lesquels peut être écrit un programme fortran :
  - ① Le format libre ;  
l'extension du fichier doit être de type ".f90" ou ultérieure.
  - ② Le format fixe (obsolète) ;  
un fichier portant l'extension ".f" ou ".for" sera considéré comme étant au format fixe.
- ▶ Pour le format libre :
  - les lignes doivent avoir une **longueur maximale** de **132 caractères** ;
  - une instruction peut s'étaler sur plusieurs lignes, en ajoutant le caractère '&' à la fin de première ligne et au début de la seconde ;
  - une ligne commençant par un '!' est ignorée (commentaire) ;

- on peut mettre plusieurs instructions par ligne en les séparant par un ';'.

► Pour le format fixe :



- les instructions doivent être dans la zone 3 (colonne 7 à 72) ;
- il ne peut y avoir qu'**une seule** instruction par ligne ;
- la zone 4 est considérée comme du commentaire ;
- la zone 1 sert à placer des étiquettes ;

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- la zone 2 sert à indiquer au compilateur que la ligne constitue la suite de la précédente (avec un '&');
- un 'C' ou un '\*' en colonne 1 indique que la ligne est du commentaire.

Fortran

Serra Sylvain

Généralités

Remarques  
Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Le nom d'une variable, fonction ou procédure peut utiliser tous les caractères alphanumériques non accentués et le blanc souligné.
  - ⚠fortran ne fait pas la différences entre les lettres minuscules et majuscules.
- ▶ Il est par contre interdit de commencer par un numéro ou le blanc souligné.

Déclarations autorisées

MaPremiereVariable  
maSecondeVariable2  
nomDeFonction\_TP

Déclarations interdites

2bad  
\_pasbon\_1  
&interdit

⚠**Rappel** : Le nom d'une variable est, d'un point de vue informatique, le nom que le programme donne à la zone de mémoire de l'ordinateur qui a été réservée pour stocker la valeur de cette variable au cours du déroulement du programme.

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Tout langage informatique comprend des opérateurs dont le but est de réaliser un certain type d'opérations.
- ▶ Chacun de ces opérateurs a un mode de fonctionnement clairement défini et ne fera que ce pour quoi il est prévu (charge au programmeur d'utiliser la bonne commande).
- ▶ Certains d'entre eux **ressemblent** à des opérateurs connus mais **ne signifient pas** du tout la même chose.  
ex : pour affecter une valeur à une variable, c'est-à-dire pour stocker cette valeur dans la mémoire de l'ordinateur, on utilise un opérateur d'affectation (ce qui reste logique) : celui-ci est quasiment toujours le signe "=", pourtant il n'a absolument rien à voir avec le symbole d'égalité mathématique.

Fortran

Serra Sylvain

Généralités

Remarques  
Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

L'ordinateur ne parle pas la même langue que nous  
Il faut donc compiler le code fortran en un fichier exécutable.  
que seul l'ordinateur comprend.



Fortran

Serra Sylvain

Généralités  
 Remarques  
 Logigramme  
 Structure  
 d'un  
 programme  
 Syntaxe  
 générale  
 Compilation

Types de  
 données

Opérateurs

Algorithmique

Sous-  
 programmes

Tableaux

Entrées –  
 Sorties

Suppléments

Informations  
 utiles

- ▶ Un code écrit en fortran n'est jamais exécuté comme tel par l'ordinateur, il doit suivre les étapes suivantes :
  - ① compilation
    - ⇒ création d'un fichier '.o' pour chaque fichier source (conservation du nom).
    - ⇒ création de fichiers '.mod' pour chaque module écrit.
  - ② édition de liens
    - Action (quasi)-invisible pour l'utilisateur.
  - ③ exécution
  
- ▶ L'ensemble de ces opérations peut se faire de deux manières :
  - ① en utilisant directement les commandes idoines ;
  - ② en utilisant des environnements de développement qui se chargent de toutes ces opérations.
    - ♣ Des choix implicites sont faits pour vous. . .

Fortran

Serra Sylvain

Généralités

Remarques

Logigramme

Structure  
d'un  
programme

Syntaxe  
générale

Compilation

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Il existe des règles de syntaxe à respecter
- ▶ Le déroulement du programme suit exactement la structure écrite (on la suit donc "avec le doigt" du début à la fin)
- ▶ Un programme doit être compilé pour obtenir un exécutable que le système d'exploitation pourra lancer

Fortran

Serra Sylvain

Généralités

Types de  
données

Dit  
simplement -  
déclaration  
Types simples  
Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

## 1 Généralités

## 2 Types de données

- Dit simplement - déclaration

- Types simples
- Types dérivés

## 3 Opérateurs

## 4 Algorithmique

## 5 Sous-programmes

## 6 Tableaux

## 7 Entrées – Sorties

## 8 Suppléments

## 9 Informations utiles

- ▶ Le langage Fortran permet de représenter l'ensemble des données usuelles : booléens, entiers, réels. . .
- ▶ Il permet en outre de "créer" des nouveaux types : coordonnées spatiales (donnée de 3 réels). . .

Le choix opéré influera sur :

- ① la place réservée en mémoire pour stocker la donnée correspondante ;
- ② l'ensemble des valeurs que pourra prendre cette variable ;
- ③ l'ensemble des opérations que l'on pourra faire avec cette variable.

## Exemple

INTEGER :: unEntier

REAL :: unreel

Fortran

Serra Sylvain

Généralités

Types de données

Dit simplement -  
déclaration

Types simples  
Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations utiles

La RAM de votre ordinateur c'est comme un chiffonnier avec plein de tiroir.

La déclaration de variable permet d'utiliser ces tiroirs

-----  
Integer :: A

-----  
Permet d'associer un tiroir (espace mémoire) à une variable nommée « A ».

On colle une étiquette pour savoir où sont rangées les infos.



Fortran

Serra Sylvain

Généralités

Types de  
données

Dit  
simplement -  
déclaration

Types simples  
Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées -  
Sorties

Suppléments

Informations  
utiles



Dans les tiroirs, on peut entrer  
différents types de variable

Integer



Real



Double precision



Logical



Complex



...

Fortran

Serra Sylvain

Généralités

Types de données

Dit simplement -  
déclaration

Types simples  
Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées -  
Sorties

Suppléments

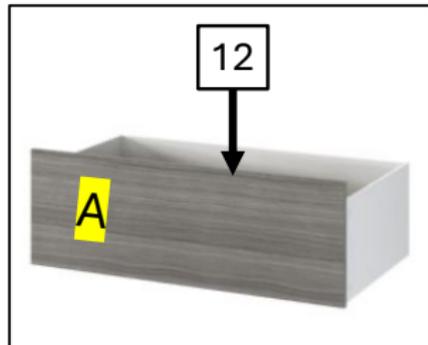
Informations utiles

Pour l'instant le tiroir est vide.

La commande

-----  
A = 12  
-----

Entre la valeur 12 dans le  
tiroir A



Fortran

Serra Sylvain

Généralités

Types de données

Dit simplement -  
déclaration

Types simples  
Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées -  
Sorties

Suppléments

Informations  
utiles

- ▶ Par défaut, tout compilateur suppose qu'une variable commençant par la lettre *i, j, k, l, m* ou *n* est de type entière et les autres de type réel.

- ▶ Ceci peut être modifié en indiquant au compilateur quelle notation implicite vous utilisez :

```
IMPLICIT INTEGER(i), REAL*8(a-h, l-z)
```

⚠ Ces façons de procéder, bien qu'autorisées, sont **extrêmement dangereuses** et sont à éviter.

- ▶ On peut (doit) aussi utiliser une autre option, qui impose au compilateur (et donc au programmeur) que toute variable soit explicitement déclarée.

Syntaxe

*IMPLICIT NONE*

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Variable pouvant prendre les valeurs `.TRUE.` ou `.FALSE.`

## Syntaxe

*LOGICAL* ,options : : unBoolean

## Exemple

```
PROGRAM Boolean
IMPLICIT NONE
! Declaration d'une variable de type logique
LOGICAL :: bool
bool = .TRUE.
PRINT*, 'Valeur de bool : ', bool
bool = .FALSE.
PRINT*, 'Valeur de bool : ', bool
END PROGRAM Boolean
```

Fortran

- ▶ Variable appartenant aux entiers relatifs.

Serra Sylvain

## Syntaxe

Généralités

*INTEGER*, options : : unEntier

Types de données

La valeur maximale autorisée dépend du nombre de bits utilisés pour stocker la variable (32, 64 ou 128).  $(-2^{n-1} \leq \text{unEntier} \leq 2^{n-1} - 1)$  ex : codage sur 32 bits  $\implies$  valeur maximale de 2 147 483 647

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

## Exemple

Opérateurs

```
PROGRAM Entier
```

Algorithmique

```
IMPLICIT NONE
```

Sous-

programmes

```
! Declaration d'une variable de type entier
```

Tableaux

```
INTEGER : : entierRelatif
```

Entrées –  
Sorties

```
entierRelatif = 123456789
```

Suppléments

```
PRINT*, 'Valeur de entierRelatif = ', entierRelatif
```

```
END PROGRAM Entier
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

**Réels**

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Variable appartenant aux réels, et donc composée de chiffres décimaux, *i.e.* d'un **point décimal**.
- ▶ Il est possible de définir **deux** types de variable réelle, en fonction de la précision voulue (nombre de chiffres significatifs) et de la place mémoire requise pour les stocker.

## Syntaxe

*REAL* ,options : : unReelSimplePrecision

*DOUBLE PRECISION* ,options : : unReelDoublePrecision

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

ex : réel simple précision codé sur 32 bits

$$1,1754944 \cdot 10^{-38} \leq |r| \leq 3,4028235 \cdot 10^{+38}$$

ex : réel double précision codé sur 64 bits

$$2.225073858507201 \cdot 10^{-308} \leq |r| \leq 1.797693134862316 \cdot 10^{+308}$$

*Remarque* : en toute rigueur, la façon de noter un réel doit correspondre avec le type de réel (`real` ou `double precision`) défini. Ainsi `3,14` doit être écrit `3.14E0` s'il s'agit d'un réel simple précision et `3.14D0` pour un double précision.

Néanmoins, les compilateurs sont aujourd'hui assez tolérants et plutôt permissifs sur ces règles de syntaxe (pensez-y toutefois en cas d'erreur sur le format d'une variable).

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

  Booléens

  Entiers

  Réels

  Complexes

  Chaînes de caractères

  Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

## Exemple

```

PROGRAM Reel
IMPLICIT NONE
! Declaration de deux variables de type reel
REAL :: reelSimple
DOUBLE PRECISION :: reelDouble
PRINT*, 'Donner la valeur de l"entier simple precision : '
READ(*,*)reelSimple
PRINT*, 'Donner la valeur de l"entier double precision : '
READ(*,*)reelDouble
PRINT*, 'Affichage des deux variables reelles'
PRINT*, ' reelSimple = ',reelSimple
PRINT*, ' reelDouble = ',reelDouble
END PROGRAM Reel

```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

  Booléens

  Entiers

  Réels

  Complexes

  Chaînes de caractères

  Quelques remarques

  Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Variable appartenant aux complexes, c'est-à-dire composée d'une partie réelle et d'une partie imaginaire.
- ▶ Le type complexe correspond ainsi à une **paire** de nombres réels (simple ou double précision).

## Syntaxe

*COMPLEX* ,options : : unComplexeSimplePrecision

*DOUBLE COMPLEX* ,options : : unComplexeDoublePrecision

Fortran

Serra Sylvain

Généralités

Types de  
données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de  
caractèresQuelques  
remarques

Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## Exemple

```
PROGRAM Complexe
IMPLICIT NONE
! Declaration de deux variables de type complexe
COMPLEX :: complexSimple
DOUBLE COMPLEX :: complexDouble
complexSimple = (1.E0,1.E0)
complexDouble = (-1.D0,-1.D0)
PRINT*,'Affichage des deux variables complexes'
PRINT*,' complexSimple = ',complexSimple
PRINT*,' complexDouble = ',complexDouble
END PROGRAM Complexe
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-

programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ Suite de caractères rangés de manière consécutive dans la mémoire de l'ordinateur.
- ▶ Ici, il faut donner la **longueur** de la chaîne de caractère.  
*Remarque* : par défaut, la valeur est 1.
- ▶ On leur assigne une valeur, *i.e.* du texte, en utilisant les apostrophes ' ' ou les guillemets " " (mais pas une combinaison des deux !)

## Syntaxe

*CHARACTER*(*LEN=ValeurEntiere*) : : uneChaineDeCaracteres

*Remarque* : autrefois (format fixe), on utilisait la notation :  
`CHARACTER*10 uneChaineDeCaracteres`

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

  Booléens

  Entiers

  Réels

  Complexes

  Chaînes de caractères

  Quelques remarques

  Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

## Exemple

```
PROGRAM ChaineCaractere
IMPLICIT NONE
! Declaration de variables de type chaine de caracteres
CHARACTER :: lettre
CHARACTER(LEN=6) :: mot
lettre = 'a'
mot = 'ensgti'
PRINT*,'La lettre est : ',lettre
PRINT*,'Le mot est : ',mot
END PROGRAM ChaineCaractere
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ On peut si on le souhaite accéder à une partie seulement d'une chaîne de caractères (on parle alors... de sous-chaîne de caractères)

## Exemple

```
PROGRAM ChaineCaractere2
IMPLICIT NONE
CHARACTER(LEN=8) :: firstName
CHARACTER(LEN=4) :: familyName
CHARACTER(LEN=4) :: nickName
firstName = "Emmanuel"
familyName = 'Kant'
nickName = firstName(3:6)
END PROGRAM ChaineCaractere2
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Toute variable déclarée peut avoir une valeur initiale.
- ▶ Il existe au moins trois méthodes pour cela (deux d'entre elles ont déjà été utilisées dans les exemples précédents) :
  - ① l'utilisation de l'opérateur d'affectation '=' ;
  - ② la saisie au clavier via la commande `READ(*,*)` ;
  - ③ l'utilisation d'une instruction `DATA`.
- ▶ Il est **impératif** d'affecter à une variable une variable de même type.  
*Remarque* : la plupart des compilateurs sont là encore assez permissifs et tolèrent quelques écarts (seulement entre entiers et réels et différents types de réels).

Fortran

Serra Sylvain

Généralités

Types de  
données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de  
caractèresQuelques  
remarques

Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## Exemple

```
PROGRAM Initialisation
```

```
IMPLICIT NONE
```

```
! Declaration des variables
```

```
INTEGER :: maVariable
```

```
! Initialisation avec l'opérateur d'affectation (i.e. "en dur")
```

```
! nb : c'est dont le PROGRAMMEUR qui décide
```

```
maVariable = 1.314D0
```

```
PRINT*,"Valeur de 'maVariable' = ",maVariable
```

```
END PROGRAM Initialisation
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de  
caractèresQuelques  
remarques

Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## Exemple

```
PROGRAM Initialisation
```

```
IMPLICIT NONE
```

```
! Declaration des variables
```

```
INTEGER :: maVariable
```

```
! Initialisation par saisie clavier
```

```
! nb : c'est dont l'UTILISATEUR qui décide
```

```
PRINT*,"Veuillez saisir un nombre entier :"
```

```
READ(*,*)maVariable PRINT*,"Valeur saisie = ",maVariable
```

```
END PROGRAM Initialisation
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de  
caractères

Quelques  
remarques

Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## Syntaxe

**DATA** variable1/valeurVariable1/, variable2/valeurVariable2/  
ou  
**DATA** variable1,variable2/valeurVariable1,valeurVariable2/

## Exemple

```
PROGRAM Initialisation
IMPLICIT NONE
! Declaration des variables
INTEGER :: var,bc
DOUBLE PRECISION :: x,y,z
DOUBLE PRECISION :: a,b
! Initialisation avec l'instruction DATA
DATA var/83/,bc/13/
DATA x,y,z/1.D0, 2.D0, 3.D0/
PRINT*,"var = ",var,"\t bc = ",bc
PRINT*,"x = ",x,"\t y = ",y,"\t z = ",z
END PROGRAM Initialisation
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de  
caractères

Quelques  
remarques

Types dérivés

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ Comme indiqué initialement, chacun des types définis peut se voir attribuer une ou plusieurs options.
- ▶ Celles qui nous intéresseront le plus seront :
  - `parameter` : pour "fixer" une variable comme constante ;
  - `dimension` : pour faire un tableau (*cf* section 8) ;
  - `allocatable` : option d'allocation dynamique de mémoire pour un tableau (*ibidem*) ;
  - `intent` : explicitation du rôle des arguments d'un sous-programme (*cf* section 6).
- ▶ Et les plus spécifiques (niveau expert) :
  - `external` : identification d'un nom comme étant celui d'un sous-programme (*ibidem*)

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Booléens

Entiers

Réels

Complexes

Chaînes de caractères

Quelques remarques

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

## Attribut PARAMETER

Il indique que la variable correspondante est **constante** durant tout le programme, et donc non modifiable.

### Exemple

```

:
```

```

LOGICAL, PARAMETER :: VRAI=.TRUE., FAUX=.FALSE.
```

```

DOUBLE PRECISION :: PI
```

```

PARAMETER (PI=3.14159265D0)
```

```

:
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Les types simples définis précédemment peuvent ne pas suffire à décrire l'ensemble des variables d'un problème.
- ▶ Dans ce cas, il est possible de définir soi-même des nouveaux objets regroupant des données de type hétérogènes : les **type** ou structure de données.
- ▶ Une structure de donnée possède un nom (à la manière de INTEGER, REAL... ) et divers attributs, appelés champs ou composantes.

## Syntaxe

*TYPE* nomDuTypeQueJeVeuxCréer

*déclaration des différents champs*

*END TYPE*

- ▶ L'accès à un champ se fait ensuite via le caractère ' % '

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

## Exemple

```
⋮  
TYPE Point  
DOUBLE PRECISION :: x,y,z  
END TYPE  
⋮
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

## Exemple

```

PROGRAM TypePoint
IMPLICIT NONE
! 1) Déclaration d'une structure pour décrire un point
TYPE Point
DOUBLE PRECISION :: x,y,z
END TYPE
! 2) Déclaration de variables de type Point
Type(Point) A,B,C
A = Point(0.D0,0.D0,0.D0)
B%x = 1.D0
B%y = B%x
B%z = B%x
C = A
PRINT*,'Point A = ',A%x,', ',A%y,', ',A%z
PRINT*,'Point B = ',B%x,', ',B%y,', ',B%z
PRINT*,'Point C = ',C%x,', ',C%y,', ',C%z
END PROGRAM TypePoint
    
```

Fortran

Serra Sylvain

Généralités

Types de données

Types simples

Types dérivés

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Des types correspondant aux valeurs que l'on veut représenter
- ▶ Une précision adaptable (en particulier pour les réels et les complexes)
- ▶ Possibilité de construire des nouveaux types, pour représenter n'importe quelle grandeur ou variable
- ▶ *Les variables se déclarent **toujours au début***  
*1<sup>ère</sup> erreur classique* (levée à la compilation)
- ▶ *Toutes les variables **doivent être initialisées***  
*2<sup>ème</sup> erreur classique* (surprise(s) pendant l'exécution. . . )

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques  
Arithmétiques  
Concaténation  
Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

1 Généralités

2 Types de données

3 Opérateurs

- Logiques
- Arithmétiques
- Concaténation
- Priorité globale

4 Algorithmique

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

8 Suppléments

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques  
Arithmétiques  
Concaténation  
Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Ils servent à manipuler entre elles des variables, et de les utiliser pour faire des estimations, calculs. . .
- ▶ Les variables intervenant dans une opération sont appelées "opérandes".
- ▶ On distingue :
  - ① les opérateurs logiques (avec opérandes logiques ou non) ;
  - ② les opérateurs arithmétiques ;
  - ③ les opérateurs de chaînes de caractères.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques

Opérandes  
logiques

Opérandes  
non logiques

Règles de  
priorité

Arithmétiques

Concaténation

Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ On peut appliquer cinq types d'opérateurs logiques aux variables logiques :
  - ① la négation `.NOT.`
  - ② la conjonction `.AND.`
  - ③ la disjonction `.OR.`
  - ④ l'équivalence `.EQV.`
  - ⑤ la non-équivalence `.NEQV.`
- ▶ Les opérandes sont de type logique.

# Opérateurs logiques : opérandes non logiques et résultats logiques

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Logiques  
Opérandes logiques

Opérandes non logiques

Règles de priorité

Arithmétiques

Concaténation

Priorité globale

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Ils servent à comparer des opérandes entières ou réelles et des chaînes de caractère.
- ▶ Il en existe six types :
  - ① strictement supérieur `.GT.` ou `>`
  - ② supérieur ou égal `.GE.` ou `>=`
  - ③ strictement inférieur `.LT.` ou `<`
  - ④ inférieur ou égal `.LE.` ou `<=`
  - ⑤ égal `.EQ.` ou `==`
  - ⑥ différent `.NE.` ou `/=`
- ▶ Les opérandes ne sont pas de type logique.

*Remarque :* pour les variables complexes, seules les deux dernières comparaisons ont un sens.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Logiques  
Opérandes logiques  
Opérandes non logiques

Règles de priorité

Arithmétiques  
Concaténation  
Priorité globale

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ En cas d'utilisation imbriquée de plusieurs opérateurs, la priorité est donnée aux :

- ① opérations contenues dans des blocs de parenthèses ( )
- ② opérations de comparaison
- ③ à la négation
- ④ à la conjonction
- ⑤ à la disjonction
- ⑥ à l'équivalence
- ⑦ à la non-équivalence

*Remarque* : il est conseillé d'éviter les combinaisons d'opération logique, afin d'éviter toute source d'erreur.

Il est primordial de bien penser initialement le(s) test(s).

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques  
Arithmétiques

Opérations  
classiques

Règles de  
priorité  
Concaténation  
Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations

- ▶ Il s'agit de toutes les opérations arithmétiques classiques :
  - puissance \*\*
  - multiplication \* et division /
  - addition + et soustraction -
- ▶ Le résultat de l'opération est toujours du type de l'opérande le plus "fort".  
INTEGER < REAL < DOUBLE PRECISION < COMPLEX

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques  
Arithmétiques  
Opérations  
classiques

Règles de  
priorité

Concaténation  
Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations

- ▶ Ce sont celles des mathématiques de base, on a ainsi par ordre de priorité :
  - ① puissance
  - ② multiplication et division
  - ③ addition et soustraction
- ▶ L'utilisation de parenthèses permet de forcer une priorité.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Logiques  
Arithmétiques  
Concaténation

Priorité globale

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Il s'agit d'une opération pour les chaînes de caractères, permettant d'assembler deux chaînes de caractères.
- ▶ L'opérateur associé est //

## Exemple

```
·  
·  
PRINT*, 'COU' //'COU'  
·  
·
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Logiques  
Arithmétiques  
Concaténation  
Priorité  
globale

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ En cas d'instruction faisant intervenir plusieurs opérateurs de types différents, la priorité est la suivante :
  - 1 Opérateurs arithmétiques
  - 2 Opérateurs de concaténation
  - 3 Opérateurs de comparaison
  - 4 Opérateurs logiques

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Structures itératives

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

1 Généralités

2 Types de données

3 Opérateurs

4 Algorithmique

- Structures conditionnelles
- Structures itératives

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

8 Suppléments

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si ...

Condition Si ... Sinon ...

Condition Si ... Sinon, Si ... Sinon ...

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées —

- ▶ Fortran permet de gérer les structures conditionnelles classiques :
  - Si ...
  - Si ... Sinon ...
  - Si ... Sinon, Si ... Sinon ...
  - Séquence
  
- ▶ À chaque fois, le programme évalue une expression logique et exécute alors **uniquement** la partie de code correspondante.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si ...

Condition Si ... Sinon ...

Condition Si ... Sinon, Si ... Sinon

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées -

- ▶ Il s'agit du cas le plus simple pour effectuer un test.

## Syntaxe

```
IF ( expressionLogique ) THEN  
code à exécuter si expression est vraie  
END IF
```

- ▶ On évalue l'expression :
  - si elle est vraie, le code qui suit est exécuté ;
  - si elle est fausse, on saute le code qui suit.

⚠ il est absolument **impératif** de fermer l'instruction avec la commande **END IF**

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si ...

Condition Si ... Sinon ...

Condition Si ... Sinon, Si ...

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées -

- ▶ On complique un peu puisqu'on choisit d'avoir toujours une action, que l'expression soit vraie ou fausse.

## Syntaxe

```
IF ( expressionLogique ) THEN  
code à exécuter si expression est vraie  
ELSE  
code à exécuter si expression est fausse  
END IF
```

- ▶ Après évaluation de l'expression :
  - si elle est vraie, le code qui suit le THEN est exécuté ;
  - si elle est fausse, on exécute le code situé après le ELSE.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si ...

Condition Si ... Sinon ...

Condition Si ... Sinon, Si ...

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées –

- ▶ On combine maintenant en autorisant l'évaluation de plusieurs expressions.

## Syntaxe

```

IF ( expression1 ) THEN
  code à exécuter si expression1 est vraie
ELSE IF( expression2 ) THEN
  code à exécuter si expression2 est vraie
ELSE
  code à exécuter sinon
END IF
  
```

- ▶ On peut mettre autant d'évaluation d'expressions que l'on souhaite.

*Remarque* : il est assez rare d'avoir besoin de plus de 2 ou 3 évaluations successives. Dans ce cas, il est certainement judicieux de repenser le problème.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si...

Condition Si... Sinon...

Condition Si... Sinon, Si...

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées —

- ▶ On teste différentes valeurs que peut prendre une variable.

## Syntaxe

*SELECT CASE ( variable )*

*CASE ( valeur1 )*

*code à exécuter si variable vaut valeur1*

*CASE ( valeur2 )*

*code à exécuter si variable vaut valeur2*

*CASE ( valeur3 )*

*code à exécuter si variable vaut valeur3*

*CASE DEFAULT*

*code à exécuter sinon*

*END SELECT*

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Condition Si...

Condition Si... Sinon...

Condition Si... Sinon, Si...

Condition Séquentielle

Condition Où (tableaux)

Structures itératives

Sous-programmes

Tableaux

Entrées -

- ▶ les "valeurs" ne sont pas nécessairement des scalaires, on peut en effet tester plusieurs possibilités en même temps.

## EXEMPLE

```

INTEGER :: mois
SELECT CASE (mois)
CASE(1,2,3)
PRINT*, 'Hiver'
CASE(4,5,6)
PRINT*, 'Printemps'
CASE(7 :9)
PRINT*, 'Ete'
CASE(10 :12)
PRINT*, 'Automne'
CASE DEFAULT
PRINT*, 'Saison inexistante...'
END SELECT
    
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nellesCondition  
Si...Condition  
Si... Sinon...Condition  
Si... Sinon, Si... SinonCondition  
SéquentielleCondition Où  
(tableaux)Structures  
itérativesSous-  
programmes

Tableaux

Entrées —

- ▶ Opération conditionnelle permettant de tester une expression logique pour **tous** les éléments d'un **tableau**.
- ▶ En conséquence, voir les fonctions pour tableaux dans la partie idoine.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ Fortran autorise aussi les structures itératives usuelles :
  - Pour ...
  - Tant que ...
  - Pour tous ...

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Structures itératives

Boucle Pour...

Boucle Tant que...

Boucle Pour Tous (tableaux)

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Commande permettant de répéter un ensemble donné d'opérations.

## Syntaxe

*DO* compteur = indiceDebut, indiceFin , increment

*code à exécuter de manière répétitive*

*END DO*

- ▶ Le nombre d'itérations est évalué au départ par le programme.
- ▶ La variable compteur peut être de type entière ou réelle (simple ou double).

*Remarque* : il est toutefois recommandé de n'utiliser que des variables entières.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## EXEMPLE

```
:\n:\nINTEGER :: comp\nDO comp = 1,10\nPRINT*, ' comp = ',comp\nENDDO
```

```
:\n:\n
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## EXEMPLE

```
:\n:\nINTEGER :: comp\nDO comp = 100,0,-10\nPRINT*, ' comp = ',comp\nENDDO
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nellesStructures  
itérativesBoucle  
Pour...Boucle Tant  
que...Boucle Pour  
Tous  
(tableaux)Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## EXEMPLE

```
 :  
 :  
 INTEGER :: comp  
 DO comp = 1000,0  
 PRINT*, ' Vais-je apparaitre '  
 ENDDO  
 :  
 :
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Structures itératives

Boucle Pour...

Boucle Tant que...

Boucle Pour Tous (tableaux)

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Il est possible de répéter plusieurs fois une lecture ou un affichage.

## Syntaxe

`CommandeEntreeSortie (ActionRépétée, compteur = indiceDebut, indiceFin , increment )`

*Remarque* : on parle de boucle implicite.

## EXEMPLE

```

:
:
INTEGER :: i
PRINT*,(i,i=1,9,2)
:
:

```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Structures itératives

Boucle Pour...

Boucle Tant que...

Boucle Pour Tous (tableaux)

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

- ▶ Répétition d'opérations jusqu'à réalisation d'une condition.

## Syntaxe

```
DO WHILE ( expressionLogique )
code à exécuter de manière répétitive
END DO
```

- ▶ Le nombre d'itérations n'est pas connu à priori.
- ⚠ **Attention** aux boucles infinies...

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## EXEMPLE

```
:  
:  
INTEGER :: N  
N = 25  
DO WHILE(N > 0)  
PRINT*, ' N = ', N  
N = N-1  
ENDDO  
:  
:
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ Il peut arriver que l'on souhaite sortir d'une boucle avant sa fin, soit parce qu'un critère critique a été atteint soit parce qu'on part dans une boucle infinie.
- ▶ Il existe pour cela plusieurs commandes utiles :
  - la commande CYCLE permet de sauter la suite du bloc itératif et de passer à l'itération suivante ;
  - la commande EXIT permet de sortir d'un bloc DO ;
  - la commande STOP permet d'arrêter complètement l'exécution du programme.  
*Remarque* : celle-ci peut être utilisée à la fin du programme pour le stopper "proprement".

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Structures conditionnelles

Structures itératives

Boucle Pour...

Boucle Tant que...

Boucle Pour Tous (tableaux)

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

## EXEMPLE

```

:
:
INTEGER :: i,j
j = -1
DO i=1,10
j = -1*j
IF(j < 0) CYCLE
PRINT*, 'i = ',i
ENDDO
:
:

```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nellesStructures  
itérativesBoucle  
Pour...Boucle Tant  
que...Boucle Pour  
Tous  
(tableaux)Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

## EXEMPLE

```
:  
:  
INTEGER :: i  
DO i=1,10  
IF(i > 7) EXIT  
PRINT*, 'i = ', i  
ENDDO  
:  
:
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nellesStructures  
itérativesBoucle  
Pour...Boucle Tant  
que...Boucle Pour  
Tous  
(tableaux)Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

**EXEMPLE**

```
·  
·  
INTEGER :: i  
DO i=1,10  
IF(i > 7) STOP  
PRINT*,i = ',i  
ENDDO  
·  
·
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ Généralisation des boucles conçue pour les **tableaux**.
- ▶ Comme précédemment voir la partie sur les tableaux.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Structures  
condition-  
nelles

Structures  
itératives

Boucle  
Pour...

Boucle Tant  
que...

Boucle Pour  
Tous  
(tableaux)

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

- ▶ On peut tester une condition et adapter le comportement du code à cette condition  
*Attention à la façon d'écrire cette condition*
- ▶ On peut répéter une opération un grand nombre de fois  
*Bien vérifier que cette répétition ait une fin*

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

**Procédures et  
fonctions**  
Fonctions  
intrinsèques  
Modules

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

1 Généralités

2 Types de données

3 Opérateurs

4 Algorithmique

5 Sous-programmes

- Procédures et fonctions
- Fonctions intrinsèques
- Modules

6 Tableaux

7 Entrées – Sorties

8 Suppléments

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Morceaux de code indépendants, pouvant être utilisés par plusieurs programmes différents.
- ▶ Ils servent à éviter la répétition d'instructions similaires, et l'introduction d'une modification (c'est-à-dire d'une erreur).
- ▶ Chaque sous-programme dispose d'un **nom** spécifique (unique) pour être appelé, c'est-à-dire utilisé.
- ▶ Un sous-programme peut admettre des **arguments**
- ▶ On distingue plusieurs types de sous-programmes :
  - les procédures (ou subroutine) et les fonctions, contenant du code exécutable ;
  - les fonctions intrinsèques (implicites), inhérentes au Fortran ;
  - les modules, qui réunissent des déclarations de variables et de sous-programmes, pouvant ensuite être ré-utilisés par d'autres programmes.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Il est *important* de savoir que :

- ▶ Le Fortran transmet l'adresse des variables (passage par référence) et non leur valeur !
- ▶ Il n'y a donc pas d'espace mémoire dédié aux arguments.
- ▶ Les variables associées peuvent soit :
  - être utilisées et restées inchangées  $\implies$  `INTENT(IN)` ;
  - être seulement modifiées  $\implies$  `INTENT(OUT)` ;
  - être utilisées et modifiées  $\implies$  `INTENT(INOUT)`.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Procédures et  
fonctions

Fonctions  
intrinsèques  
Modules

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Généralement, elles :
  - correspondent à des blocs de code conséquents ;
  - manipulent et modifient plusieurs variables du programme appelant ;
  - exécutent plusieurs actions.
- ▶ L'appel d'une procédure se fait par la commande `CALL`, suivie du nom de la procédure appelée et des éventuels arguments (entre parenthèses).

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Procédures et  
fonctions

Fonctions  
intrinsèques

Modules

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

## Syntaxe

***SUBROUTINE*** NomSubroutine(arg1, arg2, ..., argn)

*Déclaration des arguments*

*Déclaration des paramètres locaux*

*Instructions exécutables*

***END SUBROUTINE*** NomSubroutine

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Procédures et  
fonctions

Fonctions  
intrinsèques  
Modules

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

## Exemple

```
PROGRAM programmePrincipal
:
:
CALL maRoutine()
:
:
END PROGRAM programmePrincipal
:
:
SUBROUTINE maRoutine()
IMPLICIT NONE
:
:
END SUBROUTINE maRoutine
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques  
Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

**Programme principal**

! Création du programme  
Program principal

Integer :: a,b,c

b=3  
c=4




! Appel d'une subroutine  
Call subrout(a,b,c)

...



Je colle des étiquettes pour identifier les tiroirs et je définis leur type « Integer »



Je rentre la valeur « 3 » dans le tiroir « b » et « 4 » dans le tiroir « c »

J'appelle la subroutine « subrout » en lui donnant les adresses des tiroirs « a », « b » et « c »

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments  
Informations utiles

**Programme principal**

! Création du programme  
Program principal

Integer :: a,b,c

b=3  
c=4

b

3

c

4

! Appel d'une subroutine  
Call subrout(a,b,c)

Print\*, a

End program



J'affiche à l'écran la valeur de « a ».

Note : Pour l'instant, je n'ai pas encore créé la subroutine. Il faut le faire.

Dans cet exemple, la subroutine est située avant ou après le programme mais dans le même fichier nom\_du\_fichier.f90

Après la compilation, donc avant que l'ordinateur n'ai à s'en servir, le programme connaîtra donc la subroutine et pourra s'en servir.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

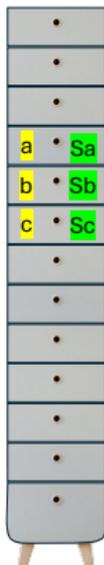
La subroutine « subrout » est définie entre les mots clés « subroutine » et « end subroutine ».

« subrout » attend 3 arguments « Sa », « Sb » et « Sc ».

Le passage d'argument permet de dire à la subroutine dans quel tiroir on a rangé les informations.

Je colle, pour la subroutine, des étiquettes « Sa », « Sb » et « Sc » aux mêmes tiroirs que « a », « b » et « c ».

La valeur 12 est entrée dans le tiroir « Sa »



## Subroutine (ou procédure)

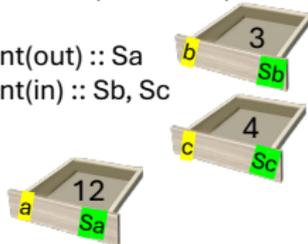
! Création du sous-programme subroutine subrout(Sa, Sb, Sc)

Integer, intent(out) :: Sa

Integer, intent(in) :: Sb, Sc

Sa = Sb\*Sc

end subroutine



« Intent(out) » = « Sa » modifiable par « subrout ».

« Intent(in) » = « Sb » et « Sc » non modifiables

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques  
Modules

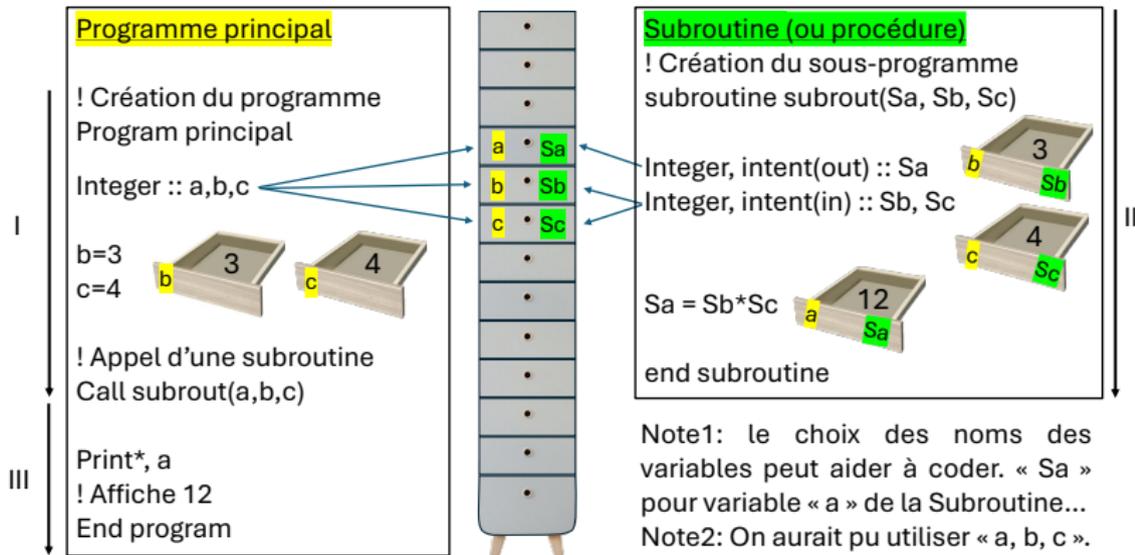
Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Ceci est la chronologie après compilation donc au moment de l'exécution, le programme connaît déjà la subroutine.



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques  
Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Elles renvoient *une valeur*.
- ▶ Elles sont associées à une variable, du nom de la fonction, stockant cette valeur.
- ▶ L'appel se fait par le nom de la fonction, suivi des éventuels arguments.

## Syntaxe

```

FUNCTION nomFonction (arg1, arg2, ..., argn)
Déclaration des paramètres (arguments transmis)
Déclaration du type retourné par la fonction
Déclaration des paramètres locaux
Instructions exécutables
nomFonction = ...
END FUNCTION nomFonction
    
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques  
Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

## Exemple

```
PROGRAM programmePrincipal
DOUBLE PRECISION :: x,y,z
DOUBLE PRECISION, EXTERNAL :: maFonctionAmoi
:
:
x = maFonctionAmoi(y,z)
END PROGRAM programmePrincipal
:
:
FUNCTION maFonctionAmoi(a,b)
DOUBLE PRECISION, INTENT(IN) :: a,b
DOUBLE PRECISION :: maFonctionAmoi
maFonctionAmoi = a+b
END FUNCTION
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques  
Modules

Tableaux

Entrées – Sorties

Suppléments

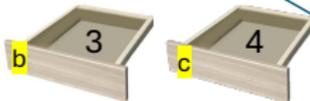
Informations utiles

## Programme principal

! Création du programme  
Program principal

Integer :: a,b,c, Fonc

b=3  
c=4



! Utilisation d'une fonction  
a = Fonc(b,c)

...



Je colle des étiquettes pour identifier les tiroirs et je définis leur type « Integer »

Je rentre la valeur « 3 » dans le tiroir « b » et « 4 » dans le tiroir « c »

J'utilise la fonction « Fonc » en lui donnant les adresses des tiroirs « b » et « c » et je rentre le résultat de la fonction dans « a »

Note : « a » et « Fonc » sont bien du même type.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

**Programme principal**

! Création du programme  
 Program principal

Integer :: a,b,c, Fonc

b=3  
c=4

b

3

c

4

! Utilisation d'une fonction  
 a = Fonc(b,c)

Print\*, a

End program



J'affiche à l'écran la valeur de « a ».

Note : Pour l'instant, je n'ai pas encore créé la fonction. Il faut le faire.

Dans cet exemple, la fonction sera une fonction externe au programme principal donc située avant ou après le programme mais dans le même fichier nom\_du\_fichier.f90

Après la compilation, donc avant que l'ordinateur n'ai à s'en servir, le programme connaîtra donc la fonction et pourra s'en servir.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

La fonction « Fonc » est définie entre les mots clés « function » et « end function ».

« Fonc » est de type « Integer ». Elle a deux arguments « Fb » et « Fc ».

Le passage d'argument permet de dire à la fonction dans quel tiroir on a rangé les informations.

Je colle, pour la fonction, des étiquettes « Fb » et « Fc » aux mêmes tiroirs que « b » et « c ».

La valeur 7 est entrée dans le tiroir « Fonc »



## Fonction

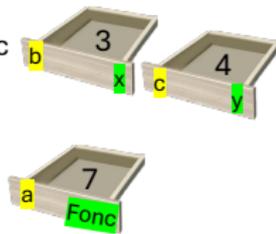
! Création de la fonction  
function Fonc(x,y)

Integer :: Fb, Fc

Integer :: Fonc

Fonc = Fa + Fb

end function



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

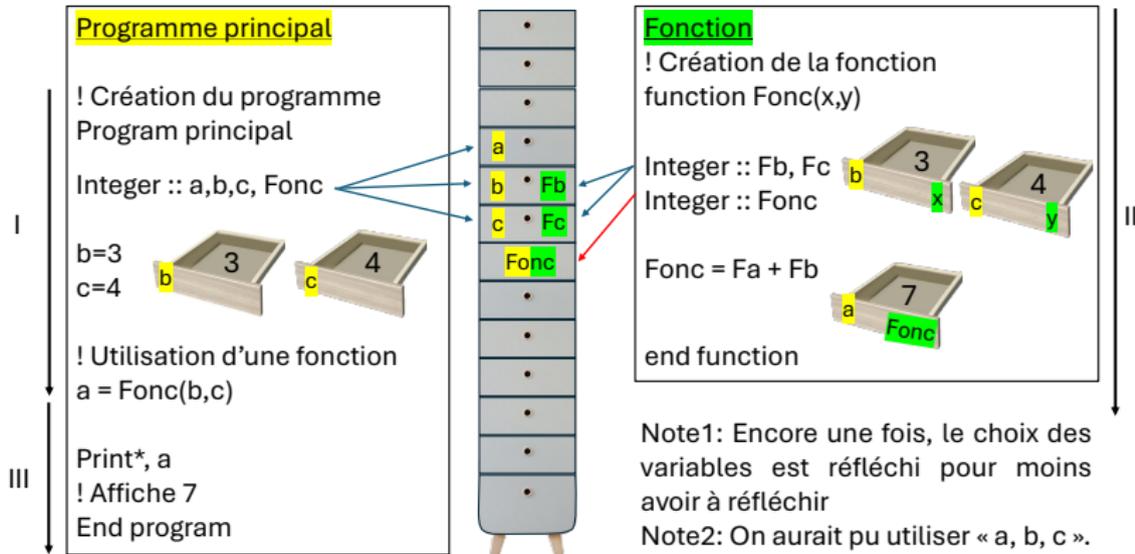
Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Ceci est la chronologie après compilation donc au moment de l'exécution, le programme connaît déjà la fonction.



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

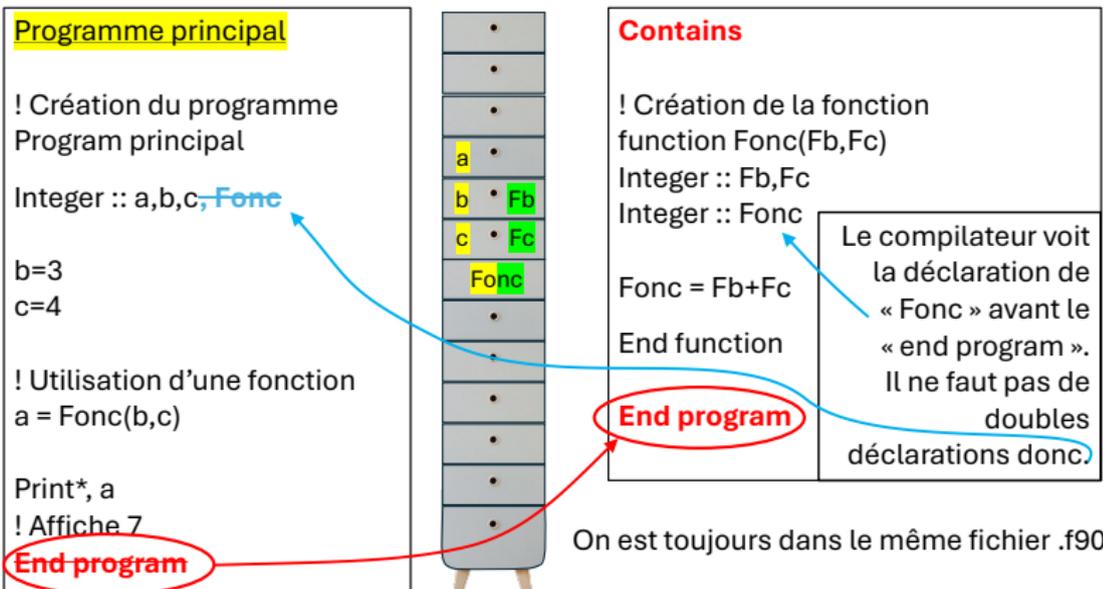
Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Exemple avec une **fonction interne**, c'est-à-dire, quand elle est définie dans le programme. Il faut alors, utiliser le mot clé « contains » et ne placer entre « contains » et « end program », que des sous-programmes (fonction ou subroutine).



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ On dispose des fonctions trigonométriques :
  - cosinus, sinus et tangente : COS, SIN et TAN  
*Remarque* : l'argument doit être exprimé en radians.
  - arc-cosinus, arc-sinus et arc-tangente : ACOS, ASIN et ATAN  
*Remarque* : le résultat est exprimé en radians.
  - cosinus, sinus et tangente hyperboliques : COSH, SINH et TANH
- ▶ Viennent ensuite les fonctions de "puissance" :
  - racine carrée : SQRT
  - exponentielle : EXP
  - logarithme népérien : LOG
  - logarithme décimal : LOG10
- ▶ Puis les fonctions de modification ou de comparaison :
  - valeur absolue : ABS

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- reste division entière : MOD

- module : MODULO

*Remarque* : exceptée la valeur absolue, les autres fonctions attendent **au moins deux** arguments.

- maximum : MAX

- minimum : MIN

*Remarque* : avec ces fonctions, on peut utiliser plus de deux arguments.

- ▶ Finalement, il reste les fonctions "complexes" :

- partie imaginaire : AIMAG

- conjugué : CONJG

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Il est assez courant de devoir convertir une variable, que ce soit :
  - ① pour manipuler des variables de type différent ;
  - ② pour approcher une variable.
  
- ▶ Les conversion de type se font via les fonctions suivantes :
  - conversion vers un entier : INT
  - conversion vers un réel simple : REAL
  - conversion vers un réel double : DBLE
  - conversion vers un complexe : CMPLX
  
- ▶ Les approximations possibles sont :
  - troncature d'un réel : AINT
  - arrondi d'un réel : ANINT
  - entier supérieur le plus proche : CEILING
  - entier inférieur le plus proche : FLOOR

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Les chaînes de caractère sont des variables un peu "spéciales", dans le sens où elles ont plusieurs éléments dont certains peuvent être des blancs.
- ▶ Il existe tout d'abord des fonctions permettant d'analyser ces chaînes :
  - longueur d'une chaîne de caractères : LEN
  - longueur d'une chaîne sans les blancs de fin : LEN\_TRIM
  - recherche d'éléments particuliers : SCAN
  - comparaison de deux chaînes : LGT et LLT, LGE et LLE
- ▶ Mais aussi des fonctions pour "retoucher" les chaînes :
  - alignement du texte à gauche : ADJUSTL
  - alignement du texte à droite : ADJUSTR
  - élimination des blancs de fin : TRIM

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions  
Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Les modules ont été développés pour généraliser les blocs communs, ils servent à incorporer dans des programmes des parties indépendantes de celui-ci.
- ▶ L'intérêt vient du fait que :
  - ① un module peut être utilisé par plusieurs programmes, on évite donc la ré-écriture de code ;
  - ② les modules permettent d'encapsuler des variables ou des sous-programmes, ce qui empêche d'autres programmeurs utilisant leur contenu de le modifier (ou tout simplement de le lire).
- ▶ Un module est enregistré dans un fichier .f90 à part de celui du programme principal.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Dans un module, il ne peut y avoir que des déclarations de variable ou des sous-programmes. Ces derniers entre les mots clés CONTAINS et END MODULE.

## Syntaxe

**MODULE** nomDuModule

*Ensemble des variables contenues dans le module*

[ CONTAINS ]

*Ensemble des procédures contenues dans le module*

**END MODULE** nomDuModule

- ▶ L'utilisation d'un module dans un programme se fait via la commande USE, qui doit précéder toutes les déclarations.
- ▶ Les règles classiques s'appliquent quant à la programmation des modules.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmesProcédures et  
fonctionsFonctions  
intrinsèques

Modules

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Au moment de la compilation, il faut utiliser le module et le programme principal.

En ligne de commande :

```
gfortran mon_module.f90 mon_programme.f90
```

Dans le logiciel CodeBlocks, il faut que les deux fichiers soient inclus dans le même projet.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions  
Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Ceci est la chronologie quand un module est utilisé. Pour rappel, les deux bouts de code ne sont pas dans les mêmes fichiers .f90

## Programme\_principal.f90

I  
↓  
! Création du programme  
Program principal  
! On utilise un module  
**Use mod\_un**

Integer :: a,b,c  
b=3  
c=4

III  
↓  
! Utilisation d'une fonction  
a = Fonc(b,c)

Print\*, a  
End program

Utiliser un module, ayant une fonction revient au même que d'avoir une fonction interne.

Il ne faut pas déclarer la variable « Fonc » dans le programme principal, car le compilateur voit la déclaration dans le module. Ici

## Module.f90

! Création du module  
**Module mod\_un**  
! déclaration de variable  
! Rien ici  
**Contains**

! Création de la fonction  
fonction Fonc(Fb,Fc)  
Integer :: Fb,Fc  
Integer : Fonc

Fonc = Fb+Fc

End function  
**End module**

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Procédures et fonctions

Fonctions intrinsèques

Modules

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Définition d'un traitement particulier dans une unité de programme qui peut être appelée à volonté
- ▶ Arguments (éventuels) passés par adresse
- ▶ Un paramètre formel n'a rien à voir avec un paramètre d'appel. . . *1<sup>ère</sup> erreur classique*
- ▶ Une fonction renvoie obligatoirement une valeur

Fortran

Serra Sylvain

Généralités

① Généralités

Types de  
données

② Types de données

Opérateurs

③ Opérateurs

Algorithmique

Sous-  
programmes

④ Algorithmique

Tableaux

Déclaration  
Initialisation  
Arguments de  
sous-  
programmes  
Manipulation

⑤ Sous-programmes

Entrées –  
Sorties

⑥ Tableaux

Suppléments

Informations  
utiles

- Déclaration
- Initialisation
- Arguments de sous-programmes
- Outils (pratiques) pour la manipulation tableaux
- ⑦ Entrées – Sorties
- ⑧ Suppléments
- ⑨ Informations utiles

- ▶ Un tableau sert à ranger des variables de même type (entier, réels, type dérivé. . .).
- ▶ On accède à un élément  $k$  d'un tableau grâce à l'opérateur  $()$ .
- ▶ Un tableau possède les caractéristiques suivantes :
  - Son **rang** correspond à son nombre de dimensions (7 max en Fortran).
  - Son **étendue** correspond aux nombres d'éléments pour une direction.
  - Son **profil** est un vecteur, avec autant de composantes que le nombre de dimensions du tableau, dont les éléments sont l'étendue du tableau pour chaque dimension
  - Sa **taille** est le produit des composantes de son profil (*i.e.* le nombre d'éléments total du tableau).
- ▶ Deux tableaux possédant le même profil sont dits **conformants**. Ils peuvent apparaître ensemble dans une expression.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Allocation statique

Allocation dynamique

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

- ▶ Nous avons vu qu'un tableau sert à stocker plusieurs variables d'un même type.
- ▶ Il faut donc réserver de la mémoire pour que l'ordinateur puisse ensuite stocker ces variables.
- ▶ On utilise l'attribut `DIMENSION` avec le **nombre d'éléments** pour chaque dimension

## Syntaxe

*TYPE*, *DIMENSION*(étendue 1, ..., étendue n) :: nomTableau

- ▶ Pour fixer l'étendue dans une dimension, on peut soit :
  - donner une constante entière (la numérotation du tableau commence alors à 1);

- donner une expression `indiceDepart:indiceFin` (la numérotation du tableau commence alors à `indiceDepart` et l'étendue vaut `indiceFin - indiceDepart`).

## Exemple

```

:
:
INTEGER, DIMENSION(5) :: TabEntier
REAL, DIMENSION(3,5) :: TabReelSimple
:
:

```

- ▶ Dans cet exemple :
  - `TabEntier` est de rang 1 et `TabReelSimple` de rang 2.
  - L'étendue de `TabEntier` et `TabReelSimple` est 5, et 3 et 5.
  - Les profils de `TabEntier` et `TabReelSimple` sont (5) et (3,5).
  - La taille de `TabEntier` est 5 et celle de `TabReelSimple` 15.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Allocation statique

Allocation dynamique

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

## Exemple

```

:
:
INTEGER, DIMENSION(-3 :3) :: TabEntier2
REAL, DIMENSION(-1 :1,4) :: TabReelSimple2
:
:

```

► Dans cet exemple :

- TabEntier2 est de rang 1 et TabReelSimple2 de rang 2.
- L'étendue de TabEntier2 est 7, et celle de TabReelSimple2 de 3 et 4.
- Profils de TabEntier2 et TabReelSimple2 : (7) et (3,4).
- Taille de TabEntier2 et TabReelSimple2 : 7 et 12.
- Le **premier indice** pour le tableau TabEntier2 est **-3** et celui de la première dimension de TabReelSimple2 est **-1** (l'autre dimension commence classiquement à 1).

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Allocation statique

Allocation dynamique

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

☢ Tous les compilateurs *ne vérifient pas* automatiquement si l'indice demandé dans un tableau est bien dans les limites de celui-ci.

Accès corrects

```
REAL, DIMENSION(5) :: X
X(1) = 1.E0
PRINT*,X(1)
```

Débordement de tableau

```
REAL, DIMENSION(5) :: X
X(6) = 1.E0
PRINT*,X(6)
```

*Remarque* : il est possible d'imposer une vérification des accès tableaux (non débordement) à la compilation et/ou à l'exécution (à utiliser seulement en mode 'debug').

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Allocation  
statique

Allocation  
dynamique

Initialisation

Arguments de  
sous-  
programmes

Manipulation

Entrées –  
Sorties

Suppléments

- ▶ Il est assez courant que l'étendue, et donc la taille, d'un tableau ne soit pas connue à priori.
- ▶ Il est donc possible d'indiquer au compilateur qu'il aura à gérer un tableau dont la taille n'est pas encore fixée (elle devra néanmoins l'être au moment où le tableau sera utilisé par le programme).
- ▶ Il suffit pour cela :
  - ① d'utiliser le caractère ':' à l'endroit où on déclare habituellement l'étendue d'une dimension
  - ② d'ajouter l'attribut `ALLOCATABLE`

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration  
Allocation statique

Allocation dynamique

Initialisation

Arguments de sous-programmes  
Manipulation

Entrées – Sorties

Suppléments

## Syntaxe

*TYPE, DIMENSION( :, ..., :), ALLOCATABLE :: nomTableau*

- ▶ **Avant** son utilisation, on pourra (dera) alors allouer dynamiquement la mémoire nécessaire grâce à la commande `ALLOCATE`.

## Syntaxe

*ALLOCATE(nomTableau(étendue1,étendue12,..) ,stat = err )*

`err` est une variable (facultative) de type entier servant à contrôler l'allocation (elle vaut 0 si tout s'est bien passé)

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Allocation  
statique

Allocation  
dynamique

Initialisation

Arguments de  
sous-  
programmes

Manipulation

Entrées –  
Sorties

Suppléments

- ▶ Après utilisation, on libèrera cette mémoire via la commande `DEALLOCATE`.

## Syntaxe

```
DEALLOCATE(nomTableau ,stat = err )
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Allocation statique

Allocation dynamique

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

## Exemple

```

:
:
! Déclaration d'un tableau dynamique
INTEGER :: errAlloc
INTEGER, DIMENSION( :), ALLOCATABLE :: TabDyn
:
:
ALLOCATE(TabDyn(5), stat = errAlloc)
:
:
DEALLOCATE(TabDyn, stat = errAlloc)
:
:

```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Allocation statique

Allocation dynamique

Initialisation

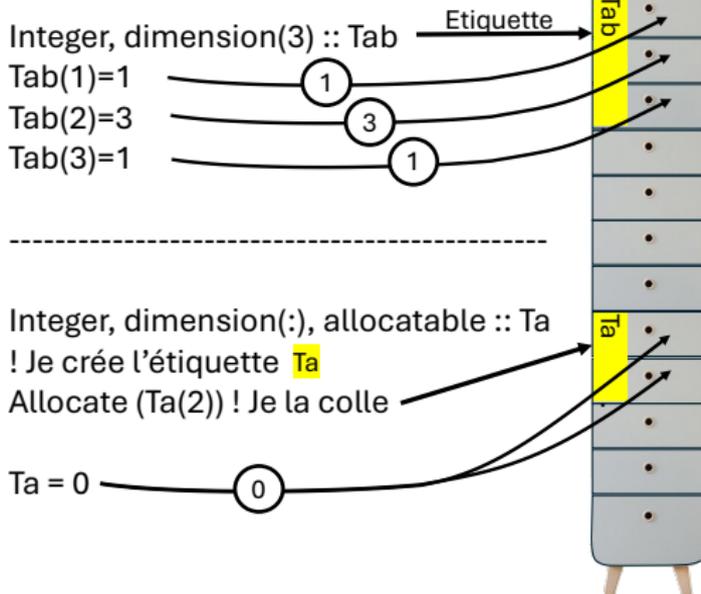
Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

Retour dans notre chiffonnier :



En fortran, par défaut, le premier indice est le 1



# Dit simplement - Tableau II

```
Integer, dimension(3,3) :: Tab
Integer :: i, j
```

2 dimensions

```
Do i = 1,3
  Do j=1,3
    Tab(i,j) = i*j
  enddo
enddo
```

```
Print*, Tab
! Affiche « 1 2 3 2 4 6 3 6 9 »
```

Comme dans un excel

	A	B	C
1	1	2	3
2	2	4	6
3	3	6	9

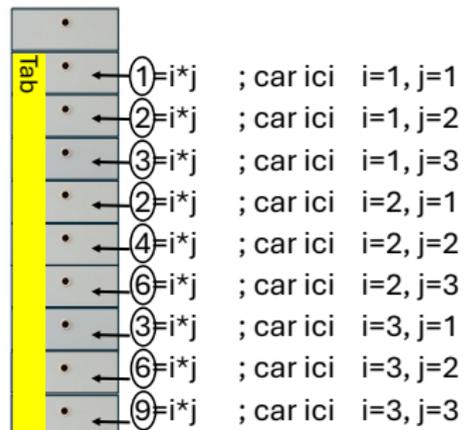


Tableau 2d, 3d, ok.  
 La 4<sup>ième</sup> dimension c'est souvent le temps  
 La 5<sup>ième</sup>, on est dans le multivers  
 6... je ne sais pas



Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Initialisation

Arguments de  
sous-  
programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Un tableau regroupant un ensemble d'éléments, il peut être initialisé de diverses manières.
- ▶ À la différence d'une variable simple, on peut initialiser :

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

**Initialisation**

Arguments de sous-programmes  
 Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- un élément unique du tableau ;

## Exemple

```

:
:
INTEGER, DIMENSION(3) :: Tab
:
:
Tab(1) = 1
Tab(2) = 2
Tab(3) = 3
:
:

```

- une portion (*i.e.* plusieurs éléments) du tableau ;

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

**Initialisation**

Arguments de sous-programmes  
Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- Pour une portion régulière, on utilise un ordre d'adressage du type `indiceDebut:indiceFin:increment`

*Remarque* : l'incrément peut être omis; il vaut alors 1.

ex :

`Tab(1:2)`  $\implies$  2 premiers éléments du tableau

`Tab(1:10:2)`  $\implies$  les éléments 1,3,5,7 et 9 (les impairs)

`Mat(1:4:1,4:2:-2)`  $\implies$  lignes 1 à 4, colonnes 4 et 2

`Mat(:,1)`  $\implies$  toutes les lignes, colonne 1 (1<sup>re</sup> colonne donc)

`Mat(1:5:2, :)`  $\implies$  lignes 1,3 et 5, toutes les colonnes
- Pour une portion irrégulière, on utilise un vecteur d'indices qu'il faut construire.

ex :

`Tab( (/1,2,3,6,8/) )`  $\implies$  les éléments 1,2,3,6 et 8

`Mat( (/1,2,6/) , (/3,7/) )`  $\implies$  lignes 1,2 et 6 des colonnes 3 et 7

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Initialisation

Arguments de  
sous-  
programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- le tableau complet.

Nous avons déjà vu que l'opérateur d'affectation peut être employé sur un tableau, il s'applique alors à tous ses éléments.

ex :  $\text{Tab} = 0 \implies$  tous les éléments sont mis à 0

On peut aussi utiliser un tableau conformant au premier.

ex :  $\text{Tab2} = \text{Tab} \implies$  tous les éléments de Tab2 auront les mêmes valeurs que les éléments de Tab

On peut ensuite utiliser le résultat d'une opération entre deux tableaux (à condition d'être toujours conformant).

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux  
Déclaration  
Initialisation

Arguments de  
sous-  
programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Les tableaux et les chaînes de caractères sont des éléments dont la taille n'est pas fixe, dans le sens où le programmeur ou l'utilisateur peuvent modifier leur taille et par conséquent la place que ces variables occupent en mémoire.
- ▶ Pour transmettre une chaîne de caractères à un sous-programme, on peut :
  - ① définir explicitement sa taille dans la procédure ;
  - ② utiliser le caractère '\*' à la place de l'attribut de longueur.
- ▶ Pour transmettre un tableau à un sous-programme, on peut :
  - ① utiliser des tableaux de taille fixe et spécifier explicitement leur taille (passage explicite rigide) ;
  - ② transmettre leur(s) étendue(s) en argument de la procédure (passage explicite ajustable) ;
  - ③ être plus subtil. . . (passage implicite).

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration  
Initialisation

Arguments de  
sous-  
programmes

Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Pour qu'une fonction renvoie un tableau, on peut :
  - ① utiliser des tableaux de taille fixe et spécifier explicitement leur taille (passage explicite rigide) ;
  - ② transmettre leur(s) étendue(s) en argument de la procédure (passage explicite ajustable) ; (pour utilisateur averti)
- ⚠ Pour une fonction, dans tous les cas, il faudra *définir une interface* qui permette au compilateur de vérifier le rôle des arguments. (pour utilisateur averti)

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

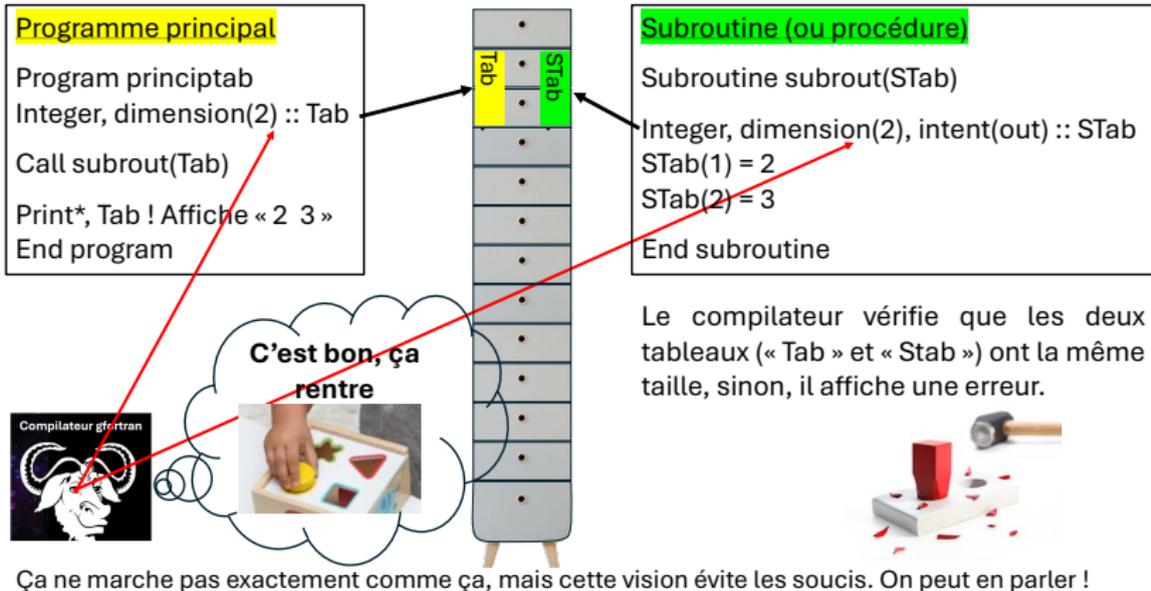
Déclaration  
Initialisation

Arguments de sous-programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations utiles



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

Informations utiles

## Programme principal

```

Program principtab
Integer, allocatable, dimension(:) :: Tab
Integer :: a
a=2
Allocate(Tab(a))
Call subrout(a,Tab)
Print*, Tab ! Affiche « 2 3 »
End program
    
```



## Subroutine (ou procédure)

```

Subroutine subrout(STab)
Integer :: Sa
Integer, dimension(Sa), intent(out) :: STab
STab(1) = 2
STab(2) = 3
End subroutine
    
```

Attention:

Si vous cherchez à mettre l'étiquette « Stab » utilisant « Sa » avant l'étiquette « Sa », ça ne passe pas.

~~Integer, dimension(Sa), intent(out) :: STab  
Integer :: Sa~~

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Un tableau contient plusieurs éléments du même type.
- ▶ Dans un ordre d'entrée/sortie, on peut donc affecter l'ensemble du tableau ou uniquement l'une de ses dimensions, voire un seul élément.
- ▶ L'utilisation de l'instruction est alors utilisée sur tout ou partie du tableau.

## Exemple

```

INTEGER, DIMENSION(3,3) :: Tab
WRITE(*,*)Tab
WRITE(*,*)Tab(:,1)
WRITE(*,*)Tab(2,2)
READ(*,*)Tab
READ(*,*)Tab(:,1)
READ(*,*)Tab(2,2)
    
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Initialisation

Arguments de  
sous-  
programmes

Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Les tableaux sont des entités contenant plusieurs variables.
- ▶ Il est courant de s'interroger sur les propriétés intrinsèques de ces entités, mais aussi d'effectuer des opérations combinant tous les éléments du tableau.
- ▶ Les fonctions d'interrogation sont :
  - profil d'un tableau : `SHAPE`
  - taille d'un tableau : `SIZE`

*Remarque* : on peut aussi obtenir l'étendue dans une direction en indiquant celle-ci en second argument

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration  
Initialisation  
Arguments de  
sous-  
programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- confirmation d'allocation : `ALLOCATED`  
*Remarque* : le tableau doit être évidemment de nature `ALLOCATABLE`...
- ▶ Les fonctions de manipulation pour les tableaux contenant des valeurs numériques sont :
  - somme : `SUM`
  - produit : `PRODUCT`
  - valeur maximale : `MAXVAL`
  - position de la valeur maximale : `MAXLOC`
  - valeur minimale : `MINVAL`
  - position de la valeur minimale : `MINLOC`

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration  
 Initialisation  
 Arguments de sous-programmes  
 Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- décompte : COUNT

*Remarque* : en indiquant une dimension en argument des fonctions précédentes, on effectue ces opérations uniquement sur les éléments de la dimension concernée.

*Remarque bis* : on peut aussi ajouter un masque (expression logique) en argument, ce qui permet de limiter l'opération aux variables respectant le masque.

- transposée : TRANSPOSE

*Remarque* : limitée aux tableaux de rang 2

- produit scalaire : DOT\_PRODUCT

- produit matriciel : MATMUL

*Remarque* : ces deux dernières fonctions requièrent **deux** arguments

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration

Initialisation

Arguments de  
sous-  
programmes

Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Opération conditionnelle de type << Où >>.
- ▶ Permet de tester une expression logique pour **tous** les éléments d'un **tableau**.

## Syntaxe

*WHERE* ( expression )

*code à exécuter si expression est vraie*

*ELSEWHERE*

*code à exécuter si expression est fausse*

*END WHERE*

⚠ À n'utiliser qu'avec des tableaux. . .

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Déclaration

Initialisation

Arguments de sous-programmes

Manipulation

Entrées – Sorties

Suppléments

Informations utiles

- ▶ Généralisation des boucles conçue pour les **tableaux**.
- ▶ Autorise l'utilisation de plusieurs conditions de contrôle et la manipulation d'éléments d'un tableau.

## Syntaxe

```
FORALL ( expression1, [expression2, expression3, ... ] )
opération à effectuer sur le tableau
END FORALL
```

## EXEMPLE

```
:
```

```
INTEGER :: i,j
```

```
INTEGER, DIMENSION(5,5) :: Mat
```

```
FORALL (i=1 :5, j=1 :5)
```

```
Mat(i,j) = i*j
```

```
END FORALL
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Déclaration  
Initialisation  
Arguments de  
sous-  
programmes  
Manipulation

Entrées –  
Sorties

Suppléments

Informations  
utiles

- ▶ Outil pour stocker plusieurs valeurs d'un même type de variables (jusqu'à sept dimensions !)
- ▶ Possibilité de manipuler tout ou partie du tableau
- ▶ Mémoire gérée statiquement (à la compilation) ou dynamiquement (à l'exécution)
- ▶ *Toujours allouer la mémoire **avant** d'utiliser un tableau dynamique*  
*1<sup>ère</sup> erreur classique*
- ▶ *Attention aux débordements (accès en dehors des limites du tableau)*  
*2<sup>nde</sup> erreur classique*
- ▶ Attention à bien indexer pour manipuler le(s) bon(s) élément(s) !
- ▶ Il existe des fonctions spécifiques (très pratiques)

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Gestion des  
fichiers  
Lecture et  
écriture

Suppléments

Informations  
utiles

1 Généralités

2 Types de données

3 Opérateurs

4 Algorithmique

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

- Gestion des fichiers
- Lecture et écriture

8 Suppléments

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Gestion des  
fichiers  
Lecture et  
écriture

Suppléments

Informations  
utiles

- ▶ Une entrée (sortie) correspond à un transfert de données entre la machine et un périphérique.
- ▶ Nous avons déjà abordé :
  - la lecture de données, saisies au clavier par l'utilisateur.  
⇒ `READ(*,*)`
  - la sortie vers l'écran, afin d'afficher des variables, du texte. . .  
⇒ `PRINT*`
- ▶ Nous allons voir que ces deux situations sont des cas particuliers d'entrée/sortie et qu'il existe d'autres possibilités de transfert d'informations entre un programme et l'extérieur.
- ▶ Un programme fortran peut lire ou écrire sur l'ensemble des périphériques usuels :
  - clavier ;
  - écran ;

Fortran

- disque ;
- fichier ;
- imprimante.

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers  
Lecture et écriture

Suppléments

Informations utiles

- ▶ Fortran connaît deux instructions exécutables pour transférer des données :
  - la lecture, via la commande READ, est un ordre d'entrée ;

## Syntaxe

*READ*(*UniteLogique*,*Format*)*Variable1*,*Variable2*,...

- l'écriture, via la commande WRITE, est un ordre de sortie.  
*Remarque* : nous allons voir que la commande PRINT est en fait une "sous-commande"...

## Syntaxe

*WRITE*(*UniteLogique*,*Format*)*Variable1*,*Variable2*,...

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Gestion des  
fichiers  
Lecture et  
écriture

Suppléments

Informations  
utiles

- ▶ L'indication d'une astérisque '\*' comme unité logique signifie qu'on souhaite accéder à l'unité logique standard pour l'ordre considéré : le clavier en entrée, l'écran en sortie.

- ▶ L'indication d'une astérisque '\*' comme format signifie que le format sera libre.

*Remarque* : le Fortran réserve habituellement des numéros pour certains périphériques ; le 0 correspond à la console ou un fichier disque, le 5 correspond au clavier et le 6 à l'écran.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers  
 Lecture et écriture

Suppléments

Informations utiles

Vous allez créer un exécutable qui s'exécute. C'est donc lui qui fait l'action.

Si vous notez " `Read(*,*)` " c'est lui qui lit. Il attend donc quelque chose à lire :

Saisie au clavier, une ligne dans un fichier...



Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

- ▶ Il sera souvent utile, et même très pratique, de lire ou d'écrire des données dans un fichier plutôt que sur le clavier ou l'écran.
- ▶ Il existe deux grandes catégories de fichiers :
  - les fichiers séquentiels ;
  - les fichiers à accès direct.
- ▶ Les fichiers séquentiels sont lus et écrits séquentiellement (!), ce qui signifie que l'on doit lire tous les enregistrements avant de pouvoir accéder à une donnée qui nous intéresse et que l'écriture se fait dans l'ordre précis des instructions fournies par le programme.
- ▶ Les fichiers à accès direct peuvent être lus et écrits dans n'importe quel ordre.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

- ▶ Avant toute chose, lecture ou écriture, la première étape consiste à **ouvrir** un (ou plusieurs) fichiers.
- ▶ On utilise pour cela la commande `OPEN`, qui permet :
  - ① de donner un numéro d'unité logique au fichier, permettant au programme de le reconnaître ;
  - ② de donner l'adresse du fichier (son emplacement sur le disque), *i.e.* son nom ;
  - ③ le statut du fichier (existe déjà, nouveau -à créer donc-, temporaire -détruit à la fin du programme-, inconnu) ;
  - ④ d'indiquer le mode de transfert souhaité (lecture, écriture, lecture et écriture) ;
  - ⑤ le type d'accès (séquentiel ou direct) ;
  - ⑥ le format des enregistrements ;
  - ⑦ de contrôler si l'ouverture s'est bien déroulée.
- ▶ L'ordre des options est indifférent.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

## Syntaxe

*OPEN(UNIT=NumeroUniteLogique, FILE=AdresseDuFichier, STATUS=StatutDuFichier, ACTION=TypeTransfert, POSITION=PositionnementCurseur, ACCESS=TypeAcces, FORM=Format, IOSTAT=ControleBonDeroulement)*

- 1 Le numéro d'unité logique doit être un **entier positif**, différent des valeurs réservées citées précédemment, qui permettra ensuite au programme d'accéder à ce fichier.

*Remarque* : pour éviter tout problème, numéroter en partant de 10...

⚠ Attention à ne pas chercher à ouvrir un fichier avec le même numéro d'unité logique...

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

☢ Le Fortran laisse le soin au programmeur de vérifier que les transferts se font dans les fichiers désirés en cas d'ouverture multiple. . .

- ② L'adresse du fichier doit être fournie sous la forme d'une chaîne de caractères, c'est-à-dire :
  - sous forme d'une constante citée entre crochets '' ou apostrophes '' ;
  - via une variable de type chaînes de caractères.
- ③ Le statut d'un fichier est fournie sous forme d'une chaîne de caractères correspondant obligatoirement à l'une des options suivantes :
  - 'OLD' : le fichier existe déjà (erreur si ce n'est pas le cas) ;
  - 'NEW' : le fichier n'existe pas et va être créé (erreur si le fichier existe) ;

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

- 'UNKNOWN' : l'existence du fichier est inconnue, il sera donc créé s'il n'existe pas et ouvert dans le cas contraire (et dans ce cas écrasé) ;
- 'APPEND' : le fichier existe et on l'ouvre en se plaçant à la fin, pour écrire à la suite de ce qu'il contient déjà ;
- 'SCRATCH' : le fichier n'est créé que durant l'exécution du programme puis détruit ensuite (fichier temporaire).

*Remarque* : l'option par défaut est 'UNKNOWN'

#### ④ On accède à un fichier de trois manières :

- 'READ' : en lecture uniquement (erreur lorsqu'on tentera d'écrire dans ce fichier) ;
- 'WRITE' : en écriture uniquement (erreur lorsqu'on tentera de lire dans ce fichier) ;
- 'READWRITE' : lecture et écriture autorisées.

*Remarque* : l'option par défaut est 'READWRITE'

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

- ⑤ Le positionnement du curseur à l'ouverture d'un fichier peut se faire de trois manières :
- 'REWIND' : retour au tout début du fichier ;
  - 'APPEND' : à la fin du fichier (on conserve donc le contenu) ;
  - 'ASIS' : dernières position en date.

- ⑥ L'accès d'un fichier se fait uniquement de deux manières :
- 'SEQUENTIAL' : accès séquentiel ;
  - 'DIRECT' : accès direct.

*Remarque* : l'option par défaut est 'SEQUENTIAL'

- ⑦ les transferts avec le fichier peuvent être de deux types :
- 'FORMATTED' : formatés ;
  - 'UNFORMATTED' : non formatés.

*Remarque* : l'option par défaut diffère selon le type d'accès, elle est 'FORMATTED' pour les fichiers séquentiels et 'UNFORMATTED' pour les fichiers directs.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
SortiesGestion des  
fichiersLecture et  
écriture

Suppléments

Informations  
utiles

- 8 le statut d'entrée/sortie doit être transférée vers une **variable** de type **entière**. Le programme affectera une valeur nulle à celle-ci si l'ouverture se passe bien et une valeur positive s'il y a eu un problème (la valeur correspond alors au type d'erreur rencontré).

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Suppléments

Informations utiles

- ▶ Tout fichier ouvert **doit** être explicitement fermé.
- ▶ A sa fermeture, on peut choisir de conserver (option par défaut) ou de détruire un fichier.

## Syntaxe

```
CLOSE(UNIT=NumeroUniteLogique  

, STATUS=StatutDuFichierApresFermeture  

, IOSTAT=ControleBonDeroulement)
```

- ▶ le statut sera soit KEEP (par défaut) pour conserver le fichier soit DELETE pour le supprimer.

L'autre option est identique à celle de la commande OPEN.

- ⚠ Fermer systématiquement les fichiers non utilisés. . .
- ⚠ Attention à bien fermer le(s) bon(s) fichier(s). . .

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Lecture

Écriture

Formats d'édition

Tableaux

Suppléments

Informations

- ▶ Nous savons désormais comment ouvrir un fichier, et nous avons déjà vu comment le faire sur les sorties standard.
- ▶ Nous allons voir maintenant comment employer pleinement la commande de lecture, c'est-à-dire avec les multiples options qu'elle propose.
- ▶ Chaque appel à la commande READ produit la lecture d'une ligne sur le périphérique.

## Syntaxe

```
READ(UNIT=UniteLogique,FMT=Format
, IOSTAT=controleLecture)Variable1,Variable2,...
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
SortiesGestion des  
fichiersLecture et  
écriture

Lecture

Écriture

Formats  
d'édition

Tableaux

Suppléments

Informations

- ▶ *UniteLogique* est le numéro d'unité logique du périphérique sur lequel on veut lire (écran, fichier, ...).
- ▶ *Format* est le format d'édition, passé sous la forme d'une chaîne de caractères :
  - soit directement dans l'instruction de lecture, encadré par deux apostrophes '' ;
  - soit grâce à une variable de type chaînes de caractères ;
  - soit grâce à une étiquette renvoyant à une déclaration `FORMAT`.

*Remarque* : l'astérisque indiquera un format libre.

- ▶ *controleLecture* est une variable entière permettant de suivre le déroulement de la lecture.

Lecture sans problème  $\implies$  valeur nulle

Fin du fichier atteinte  $\implies$  valeur négative

Erreur de lecture  $\implies$  valeur positive

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Lecture

Écriture

Formats d'édition

Tableaux

Suppléments

Informations

- ▶ L'écriture est produite par l'appel de la commande WRITE.

## Syntaxe

```
WRITE(UNIT=UniteLogique,FMT=Format
, IOSTAT=controleLecture
, ADVANCE=OuiNon)Variable1,Variable2,...
```

- ▶ Les options sont identiques à celle de la commande READ.
- ▶ L'option ADVANCE permet d'indiquer comment se fera la prochaine écriture :
  - valeur 'YES' : sur une nouvelle ligne (par défaut);
  - valeur 'NO' : à la suite de la sortie actuelle.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
SortiesGestion des  
fichiersLecture et  
écriture

Lecture

Écriture

Formats  
d'édition  
Tableaux

Suppléments

Informations

- ▶ Dans le cas particulier où l'on veut écrire sur le périphérique de sortie standard, on peut utiliser la commande PRINT que nous avons déjà vu.

## Syntaxe

*PRINT* *Format*, Variable1, Variable2, . . .

L'utilisation de l'astérisque indiquera encore une fois un format d'édition libre.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Lecture  
Écriture

Formats d'édition

Tableaux

Suppléments

Informations

- ▶ Un format d'édition fournit au programme une "méthode" de lecture ou d'écriture des variables.
- ▶ Le format libre se spécifie par l'astérisque \*.
- ▶ Tout autre format est défini par la donnée de descripteurs, décrivant le type de variables qui va être utilisé.

Ces descripteurs sont :

- L pour une variable booléenne ;
- I pour une variable entière ;
- E et D ou F pour une variable réelle (notation scientifique -réel simple ou double- ou décimale) ;  
*Remarque* : il faut aussi dans ce cas préciser la partie décimale.  
*Remarque 2* : pour la notation scientifique, si l'on veut  $n$  chiffres après la virgule, il faut prévoir  $n + 7$  caractères au minimum
- A pour une variable caractère ;

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Gestion des  
fichiers

Lecture et  
écriture

Lecture  
Écriture

Formats  
d'édition

Tableaux

Suppléments

Informations

- X pour un blanc ou un espace.
- ▶ Chaque descripteur est suivi d'un chiffre indiquant le nombre de caractères à consacrer pour la variable.

## Exemple

```
WRITE(*,'(L5)')
```

```
WRITE(*,'(I8)')
```

```
WRITE(*,'(E15.8)')
```

```
WRITE(*,'(D23.16)')
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Lecture

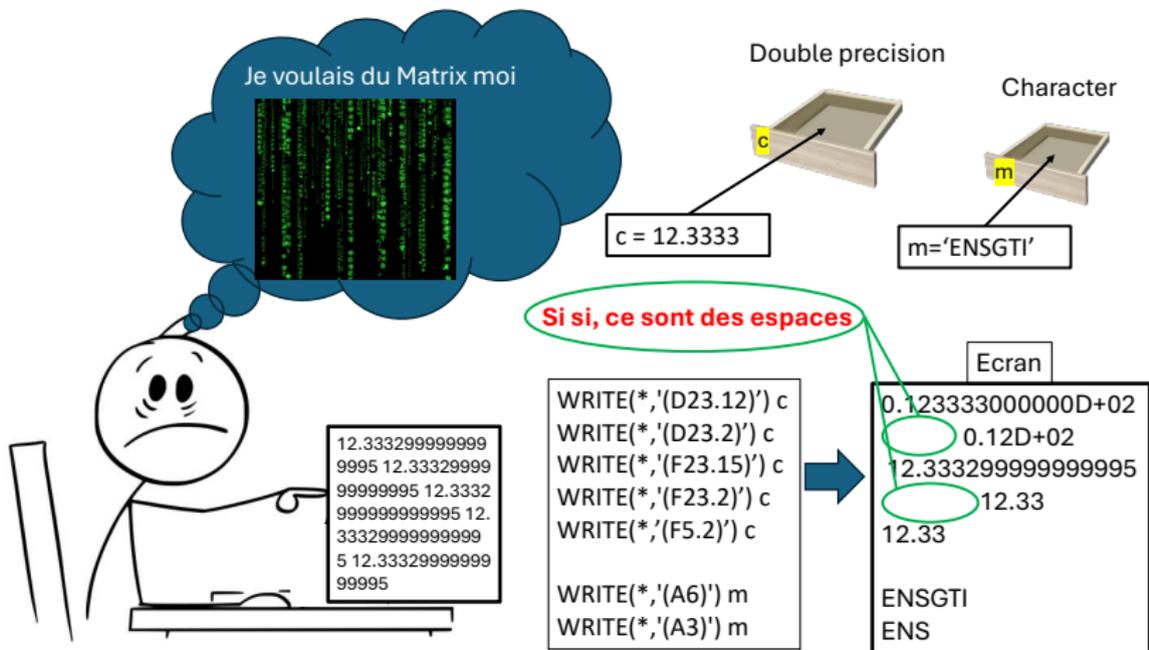
Écriture

Formats d'édition

Tableaux

Suppléments

Informations



Je voulais du Matrix moi

Double precision

Character

`c = 12.3333`

`m = 'ENSGTI'`

**Si si, ce sont des espaces**

```

WRITE(*,'(D23.12)') c
WRITE(*,'(D23.2)') c
WRITE(*,'(F23.15)') c
WRITE(*,'(F23.2)') c
WRITE(*,'(F5.2)') c

WRITE(*,'(A6)') m
WRITE(*,'(A3)') m
    
```

Écran

```

0.1233333000000D+02
0.12D+02
12.3332999999999995
12.33
12.33

ENSGTI
ENS
    
```

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Gestion des  
fichiers

Lecture et  
écriture

Lecture

Écriture

Formats  
d'édition

Tableaux

Suppléments

Informations

- ▶ Un tableau contiendra **plusieurs** valeurs ;
- ▶ Il faudra donc faire une lecture (ou écriture) pour chacune !

Plus d'informations dans la partie sur les tableaux...

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Gestion des fichiers

Lecture et écriture

Lecture

Écriture

Formats d'édition

Tableaux

Suppléments

Informations

- ▶ On utilise des flots, pour les entrées et/ou les sorties depuis le programme
- ▶ Pour les fichiers, le *chemin doit être fourni*
- ▶ Le code peut obtenir beaucoup d'autorisations du programmeur...  
*Attention à ne pas tout casser!*
- ▶ Les valeurs lues et/ou sorties doivent être associées à des variables d'un type adéquat
- ▶ Il existe des formats d'écriture et de lecture

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

1 Généralités

2 Types de données

3 Opérateurs

4 Algorithmique

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

8 Suppléments

- Nouvelles possibilités pour les sous-programmes
- Manipulation des tableaux
- Options de lecture/écriture
- Interface
- Utilisation de commandes externes au programme
- Précision des variables
- Portabilité et visibilité des variables

9 Informations utiles

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Sous-  
programmes

Tableaux

Fichiers

Interface

Commandes  
OS

Précision

Portabilité et  
visibilité des  
variables

- ▶ Il est possible de ne plus passer certains arguments à un sous-programme.
- ▶ Cela permet d'éviter le passage systématique de variables qui ne sont pas nécessairement utiles à chaque utilisation de la procédure.
- ▶ On ajoute alors l'option `OPTIONAL` à l'argument considéré.
- ▶ La présence effective de cet argument est alors testée avec la commande `PRESENT`.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ On peut inclure des sous-programmes dans un programme principal, contrairement à ce qui se faisait jusqu'alors.
- ▶ La distinction est alors que :
  - les sous-programmes internes font partie intégrante de l'unité de programmation qui les contient (programme principal ou sous-programme). Concrètement, cela signifie qu'ils ont accès à toutes les variables que l'unité contient, celles-ci étant alors considérées comme des variables **globales** (qu'elles soient locales ou globales dans l'unité contenant).  
Localement, si une variable **locale** d'un sous-programme interne est déclarée avec le même nom qu'une variable globale, celle-ci devient **masquée** ; sa valeur est conservée mais elle devient inaccessible, la variable prenant alors la valeur définie dans le sous-programme.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- les sous-programmes externes sont indépendantes des autres entités (ce sont des unités de compilation séparées), aucune variable n'est partagée et ne peut donc être connue si elle n'est pas transmise en tant qu'argument.

Pour faire simple, rien de ce qui est connu à un endroit ne l'est à un autre ; on peut donc avoir des variables ayant le même nom mais des valeurs différentes. Le seul moyen de communiquer des informations passe par le passage de variables comme arguments de sous-programmes.

*n.b.* c'est ce que vous faisiez jusqu'à maintenant !

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

## Syntaxe

**PROGRAM** NomProgrammePrincipal

*Instructions du programme principal*

**STOP**

**CONTAINS**

*Définition des sous-programmes internes*

**END PROGRAM** NomProgrammePrincipal

*Définition des sous-programmes externes*

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Sous-  
programmes

Tableaux

Fichiers

Interface

Commandes  
OS

Précision

Portabilité et  
visibilité des  
variables

- ▶ Il peut arriver qu'une procédure ait pour paramètre un argument de type procédure.
- ▶ Il faut alors le préciser au programme, via l'option `EXTERNAL`, au moment de la déclaration de la procédure.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision  
Portabilité et visibilité des variables

## Exemple

```

FUNCTION fonction1(x)
  REAL :: x
  REAL :: fonction1
  ...
  fonction1 = ...
  ...
END FUNCTION fonction1

...
FUNCTION fonction2(x, fonction1)
  REAL :: x
  REAL, EXTERNAL :: fonction1
  REAL :: fonction2
  ...
  fonction2 = ...
  ...
END FUNCTION fonction2
    
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision  
Portabilité et visibilité des variables

- ▶ Il est désormais possible d'envisager des sous-programmes capables de s'appeler eux-mêmes.
- ▶ Il faut néanmoins **informer** le compilateur d'une procédure (mot clef `RECURSIVE`)

## Syntaxe

*RECURSIVE SUBROUTINE* nomSubroutine (arguments)

## Syntaxe

*RECURSIVE FUNCTION* nomFonction (arguments) *RESULT*  
(nomResultat)

⚡ Pour les fonctions, il faut aussi déclarer le résultat, qui doit avoir un nom **différent** de la fonction.

# Conservation de la valeur d'une variable entre plusieurs appels

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ Une variable interne à une procédure ou une fonction peut voir sa valeur conservée entre deux appels successifs.
- ▶ Cela permet d'éviter le passage de cette variable en tant qu'argument.
- ▶ Cela permet aussi de savoir combien de fois la procédure ou fonction a été appelée.
- ▶ On ajoute alors l'option `SAVE` à l'argument considéré.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ De nombreuses fonctions ont été développées pour faciliter et optimiser la manipulation des tableaux.
- ▶ La plupart des actions proposées, comme toujours, sont aussi faisables "à la main".
- ▶ Les fonctions particulièrement intéressantes sont :
  - ALL
  - ANY
  - COUNT
  - PACK
- ▶ Là encore, ces fonctions peuvent s'utiliser avec une restriction à l'une des dimensions du tableau.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ On peut revenir en arrière d'une ligne ou tout au début d'un fichier, à l'aide des fonctions `BACKSPACE` ou `REWIND`

**Syntaxe**

*`BACKSPACE(UNIT=numeroUniteLogique)`*

**Syntaxe**

*`REWIND(UNIT=numeroUniteLogique)`*

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ Dans la manipulation de fichiers, nous avons vu que nous pourrions être amenés à lire des fichiers existants ou à en créer des nouveaux.
- ▶ Toute tentative d'ouvrir un fichier inexistant ou de créer un fichier existant conduit à une erreur (à moins d'utiliser une option spécifique).
- ▶ Pour s'assurer de ne pas faire d'impair, le Fortran propose donc une fonction permettant d'interroger un fichier et de savoir s'il existe et, dans ce cas, de connaître les options fournies à son ouverture.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

## Syntaxe

*INQUIRE (FILE=nomDuFichier ,options...)*

ou

*INQUIRE (UNIT=numeroUniteLogique ,options...)*

- ▶ On peut associer des options qui retourneront des valeurs dépendantes des caractéristiques du fichier interrogé :
  - **EXIST** : booléen vrai si le fichier existe et faux sinon ;
  - **OPENED** : booléen vrai si le fichier est ouvert et faux sinon ;
  - **NAMED** : booléen vrai si le fichier a un nom et faux sinon
  - **NUMBER** : entier contenant le numéro d'unité logique du fichier (-1 s'il n'est pas connecté) ;
  - **NAME** : chaîne de caractères contenant le nom du fichier (valeur indéfinie s'il n'a pas de nom)

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- **READ** : chaîne de caractères contenant YES (NO) si l'accès en lecture est permis (interdit) ;
  - **WRITE** : chaîne de caractères contenant YES (NO) si l'accès en écriture est permis (interdit)
  - **READWRITE** : chaîne de caractères contenant YES (NO) si l'accès en lecture et écriture est permis (interdit) ;
  - **ACTION** : chaîne de caractères contenant le mode d'ouverture ;
  - **POSITION** : chaîne de caractères contenant le positionnement à l'ouverture ;
  - **ACCESS** : chaîne de caractères contenant le type d'accès ;
  - **FORM** : chaîne de caractères contenant le format d'ouverture  
*Remarque* : les options précédentes auront une valeur indéfinie si le fichier n'est pas connecté.
- ▶ Enfin, on peut aussi utiliser les options classiques afin de contrôler le bon déroulement de l'instruction d'interrogation.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision  
Portabilité et visibilité des variables

- ▶ Afin d'éviter l'utilisation d'un format d'édition et la donnée de nombreuses variables, il est possible de regrouper ces variables afin de faciliter leur lecture et écriture.
- ▶ On utilise pour cela une liste :

## Syntaxe

*NAMELIST* / nomListeVariables / ListeDesVariables

- ▶ Lors de toute entrée ou sortie, on remplace l'utilisation de la commande `READ` ou `WRITE` utilisée avec un format d'édition et une liste de variables par une instruction de liste :
  - la liste commence par `&nomListe` ;
  - chaque variable est séparée par une virgule `,` ;
  - la fin de liste est indiquée par le symbole `/`.

*N.B.* : cela permet aussi de revenir en arrière en cas d'erreur.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ Une interface peut servir à deux choses principales
  - ① l'organisation de code source ;
  - ② la surcharge d'opérateur.
- ▶ Elle est définie par un nom spécifique et peut contenir :
  - des procédures, fonctions ou sous-programmes ;
  - des variables d'environnement ;

*N.B.* : les interfaces se situent préférentiellement dans un module.

## Syntaxe

```
INTERFACE nomInterface
:
:
END INTERFACE nomInterface
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision  
Portabilité et visibilité des variables

- ▶ Une interface peut servir à surcharger un opérateur, c'est-à-dire à étendre cet opérateur à des types non triviaux et à définir le comportement associé.
- ▶ Cela permet aussi d'envisager de nouveaux opérateurs.

## Syntaxe

```

INTERFACE OPERATOR (+)
MODULE PROCEDURE somme
END INTERFACE
:
:
FUNCTION somme(P1,P2)
:
:
END FUNCTION somme
    
```

⚠ l'opérateur d'affectation se surcharge différemment. Il faut utiliser l'instruction **ASSIGNMENT** au lieu de **OPERATOR**.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Sous-  
programmes

Tableaux

Fichiers

Interface

Commandes  
OS

Précision  
Portabilité et  
visibilité des  
variables

- ▶ Il est possible d'appeler à travers un programme Fortran des commandes du système d'exploitation.
- ▶ Au-delà de cela, on peut aussi interroger l'horloge du processeur.

☢ Ceci est très dépendant du type d'OS...

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ Pour rappel, la précision de chaque type proposé par Fortran (entier, réels. . .) dépend du compilateur.  
Habituellement, le nombre d'octets utilisé est ainsi :
  - pour les entiers : **4** octets
  - pour les réels (simple précision) : **4** octets
  - pour les réels (double précision) : **8** octets
- ▶ Il est possible de modifier cette précision et d'indiquer au compilateur celle qui est désirée en remplacement

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

- ▶ On utilise pour cela le paramètre `KIND`

## Exemple

`INTEGER(KIND=1) :: entierCodeSurUnOctet`

`INTEGER(KIND=8) :: entierCodeSurHuitOctets`

`DOUBLE PRECISION(KIND=16) ::`

`reelDoubleCodeSurSeizeOctets`

- ▶ On peut en outre laisser le compilateur adapter la précision (et donc éviter d'indiquer le nombre d'octets souhaités) de manière à représenter des nombres en respectant un certain intervalle.
- ▶ On utilise pour cela les fonctions `SELECTED_INT_KIND` et `SELECTED_REAL_KIND`.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Sous-programmes

Tableaux

Fichiers

Interface

Commandes OS

Précision

Portabilité et visibilité des variables

## Syntaxe

*INTEGER(SELECTED\_INT\_KIND (n))* :: varEntiere

où  $n$  est un entier tel que la variable entière aura une précision lui permettant d'être comprise entre  $-10^n$  et  $10^n$ .

## Syntaxe

*DOUBLE PRECISION(SELECTED\_REAL\_KIND (m,n))* :: varReelle

avec ici  $n$  et  $m$  tels que le réel sera compris entre  $-10^n$  et  $10^n$  et aura  $m$  chiffres après la virgule.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Sous-  
programmes

Tableaux

Fichiers

Interface

Commandes  
OS

Précision

Portabilité et  
visibilité des  
variables

- ▶ Par défaut, **toutes** les variables d'un programme sont connues des sous-programmes internes de celui-ci.
- ▶ Ceci pose plusieurs difficultés :
  - ① risque de modification involontaire d'une variable ;
  - ② limitation dans le nom d'une variable ;
  - ③ grand nombre de variables, pas nécessairement utiles dans chaque sous-partie.
- ▶ Une première solution avait été abordée via les modules et les procédures externes.
- ▶ La notion de portabilité étend ce comportement, elle ajoute un attribut à tout type de variable : **PUBLIC** ou **PRIVATE**
- ▶ Ceci permet d'**encapsuler** certaines variables, les variables privées n'étant connues qu'à l'intérieur du bloc qui les définit.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Installation

Terminal

Règles

Erreurs  
courantes

Webographie

1 Généralités

2 Types de données

3 Opérateurs

4 Algorithmique

5 Sous-programmes

6 Tableaux

7 Entrées – Sorties

8 Suppléments

9 Informations utiles

- Installation
- Utilisation du terminal
- Les dix commandements
- Erreurs courantes
- Webographie

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

- ▶ Pour programmer, vous aurez besoin d'un compilateur qui permette de traduire votre code en langage machine. Celui-ci va "expliquer" à l'ordinateur ce que vous voulez faire et construire un fichier exécutable ; contrairement au code source, celui-ci est dépendant de la plateforme et du système d'exploitation : PC avec Windows 10 ou PC avec Linux, MacBookAir avec iOS X Yosemite. . .
- ▶ Il existe des dizaines de compilateur, avec leurs spécificités propres, mais à notre niveau n'importe quel compilateur gratuit fera l'affaire ! C'est pourquoi, nous recommandons d'utiliser gfortran. Pour le trouver et l'installer, il suffit de lancer une recherche google du type `gfortran windows` ou `gfortran mac`, selon le type d'OS que vous utilisez, en précisant éventuellement la version (Windows Vista, Windows 7, Windows 10. . .). Ensuite, suivez les instructions !

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

- ▶ L'édition d'un programme, *i.e.* l'écriture du code, peut se faire avec n'importe quel outil de traitement de texte de base tels que le BlocNotes ou WordPad (sur PC) ou TextEdit (sur Mac).
  - ⚠ Ne surtout pas utiliser Word ou équivalent (OpenOffice *etc.*) car ils ajoutent des caractères cachés, notamment pour la présentation du texte, qui ne seraient pas compris par le compilateur à la compilation !
- ▶ Vous pouvez aussi utiliser un éditeur syntaxique, c'est-à-dire un logiciel qui "connaît" les langages de programmation, et les mots clefs associés, et va donc colorer votre code afin de le rendre plus lisible. Souvent, il vous permettra aussi d'organiser votre programme ou bien encore de réduire certaines parties... Il en existe là aussi des dizaines mais nous vous recommandons soit CodeBlocks (Windows et Linux) soit TextWrangler (Mac), qui sont gratuits et que vous trouverez facilement (google is your friend).

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

Action désirée	UNIX	DOS
Connaître le répertoire courant	<code>pwd</code>	<code>chdir</code>
Listing d'un répertoire	<code>ls</code>	<code>dir</code>
Créer un répertoire		<code>mkdir</code>
Changer de répertoire		<code>cd</code>
répertoire parent <code>../</code>		
répertoire courant <code>.</code>		
Afficher le contenu d'un fichier	<code>type</code>	<code>cat</code>
Copier un fichier	<code>cp</code>	<code>copy</code>
Déplacer un fichier	<code>mv</code>	<code>move</code>
Détruire un fichier	<code>rm</code>	<code>del</code>
Rafraîchir l'affichage	<code>clear</code>	<code>cls</code>
Lancer un exécutable	<code>./NomExe</code>	<code>NomExe.exe</code>

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Installation

Terminal

Règles

Erreurs  
courantes

Webographie

```
gfortran fichier.f90
```

Compile fichier.f90 pour créer l'exécutable nommé "a"

```
gfortran fichier.f90 -o NomExe
```

Compile fichier.f90 pour créer l'exécutable nommé "NomExe"

```
gfortran fichier1.f90 fichier2.f90 -o NomExe
```

Compile fichier1.f90 et fichier2.f90 pour créer l'exécutable nommé "NomExe"

```
gfortran *.f90 -o NomExe
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

Compile TOUS les fichiers .f90 pour créer l'exécutable nommé "NomExe"

```
gfortran fichier.f90 -o NomExe -fbounds-check
```

Compile fichier.f90 pour créer l'exécutable nommé "NomExe" qui vérifie systématiquement que tous les éléments de tableau manipulés existent bien (non-débordement)

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Installation

Terminal

Règles

Erreurs  
courantes

Webographie

- ① Toujours coder en mode **IMPLICIT NONE**.
- ② Toujours déclarer les variables en début de programme (idem pour les sous-programmes, *i.e.* pour les procédures et fonctions).
- ③ *Toujours initialiser* toutes les *variables*.  
Il faut donc leur affecter une valeur d'une manière ou d'une autre...
- ④ Toujours allouer la mémoire pour un tableau *avant* sa première utilisation.  
La commande ALLOCATE doit donc être utilisée avant de stocker quoi que ce soit dans ce tableau !
- ⑤ *Toujours définir* les *rôles des arguments* dans les sous-programmes.  
On utilise pour cela l'attribut INTENT...

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

- ⑥ *Toujours préciser* le *statut* et le *type d'accès* prévu lors de l'ouverture d'un *flot* (fichier).

On utilise pour cela les attributs STATUS et ACTION...

- ⑦ Toujours inclure les modules avant les déclarations

- ⑧ Fermer explicitement un flot ouvert (fichier) lorsqu'on en a plus besoin.

On utilise pour cela la commande CLOSE...

- ⑨ Désallouer explicitement la mémoire réservée pour un tableau dynamique lorsqu'on en a plus besoin.

On utilise pour cela la commande DEALLOCATE...

- ⑩ Vérifier la bonne réalisation des actions demandées, particulièrement pour l'allocation dynamique de mémoire (tableaux) et pour l'ouverture et la fermeture des flots (fichiers).

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

Nous allons résumer ici une liste (non-exhaustive) des erreurs les plus répandues :

- 1 *Oubli de déclaration* d'une variable

```
i = 0
```

```
1
```

Error: Symbol 'i' at (1) has no IMPLICIT type

La variable i n'a pas été déclarée.

- 2 Redondance d'un nom de variable (*i.e.* utilisation d'un nom pour une variable d'un nom déjà utilisé pour une autre variable)

```
REAL :: i
```

```
1
```

Error: Symbol 'i' at (1) already has basic type of

```
INTEGER
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

On cherche à déclarer une variable réelle avec le nom `i` alors qu'une variable porte déjà ce nom (ici, on nous dit que cette dernière est de type entier).

### ③ Déclaration d'une variable mal-placée

```
INTEGER toto
```

```
1
```

**Error: Unexpected data declaration statement at (1)**  
 Une déclaration a été faite dans le corps du programme alors que les déclarations doivent toujours se faire au début du programme ("*always at the beginning*").

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

- 4 Non-correspondance entre le type d'une variable et une opération ou affectation sur cette variable

Le piège, c'est qu'il n'y a aucun problème à la compilation mais on a certainement une perte d'informations (troncature très souvent des nombres réels).

- 5 *Oubli d'initialisation* d'une variable

Cela ne pose aucun problème à la compilation, pourtant l'exécution risque soit de planter soit de conduire à un calcul faux car soit la valeur 0 est assignée à cette variable (avec quelques rares - compilateurs) soit la partie mémoire correspondante est transformée en une valeur cohérente pour le type de variables.

⚠ Pour rappel, il faut toujours initialiser toutes les variables d'un programme.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

## 6 Structures conditionnelles ou boucles non fermées

Error: Expecting END IF statement at (1)

ou

Error: Expecting END DO statement at (1)

Vous avez une structure IF ou une boucle DO qui ne se termine jamais : il manque un END IF ou un END DO quelque part.

## 7 Mauvaise manipulation de tableaux :

### ■ *Débordement*

```
tab(0) = 0
```

```
1
```

Warning: Array reference at (1) is out of bounds

On cherche à affecter une valeur à l'élément indicé 0 du tableau tab or cet indice n'existe pas (0 est en dehors des limites du tableau).

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

Fortran runtime error: Array reference out of bounds for array 'tab2', upper bound of dimension 1 exceeded (in file 'Erreur.f90', at line 21)

Bien que le code soit compilé sans problème, l'exécution (avec l'option `-fbounds-check`) montre que l'on a une erreur à la ligne 21 où l'on cherche un élément dans le tableau `tab2` avec un indice supérieur à l'indice maximal.

n.b. : si dans le message, on a `lower` au lieu de `upper`, le problème se situe au début du tableau, sur son indice minimal

- Affectation ou calcul *non-conformant*

```
x = tab
```

```
1
```

```
Error: Incompatible ranks 0 and 1 in assignment at (1)
```

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

On utilise dans une affectation un scalaire (de rang 0), la variable `x`, et un tableau (de rang 1 ici), la variable `tab` ; ce qui n'est pas possible !

- Allocation dynamique non-conformante

```
ALLOCATE(tab2(5,5))
```

1

Error: Rank mismatch in array reference at (1) (2/1)

- Désallocation non faite

Cette erreur affecte surtout le temps d'exécution et la place disponible en mémoire (toutefois, il vaut mieux éviter de la faire).

## 8 Gestion des flots

- Ouverture non sécurisée

Ce n'est pas une erreur en soit, si l'on sait exactement ce que l'on fait. . . Toujours utiliser des gardes-fous et spécifier le statut du fichier et l'action qu'on va faire dessus.

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Installation

Terminal

Règles

Erreurs  
courantes

Webographie

- Ouverture consécutive avec le même numéro d'unité logique  
A priori, c'est la dernière affectation qui prime donc seul le dernier flot sera concerné par des actions de lecture et d'écriture, le (ou les) autre(s) flux ne sont donc plus accessibles.
- *Erreur de lecture*  
Fortran runtime error: Bad integer for item 1 in list  
input  
Durant la lecture dans un fichier, on a cherché à affecter une valeur non valide à l'item 1, qui est une variable de type entier. Souvent, en regardant le fichier, on s'aperçoit qu'on a une valeur décimale. . .

## 9 Modules

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

- Fichier de module non fourni

```
use unModule
```

```
1
```

Fatal Error: Can't open module file 'unmodule.mod' for reading at (1): No such file or directory

Lors de la compilation, le nom du fichier contenant la définition du module unModule n'a pas été fourni.

- Mauvaise inclusion

```
use unModule
```

```
1
```

```
Erreur.f90:2.13:
```

```
IMPLICIT NONE
```

```
2
```

Error: USE statement at (1) cannot follow IMPLICIT NONE statement at (2)

Fortran

Serra Sylvain

Généralités

Types de  
données

Opérateurs

Algorithmique

Sous-  
programmes

Tableaux

Entrées –  
Sorties

Suppléments

Informations  
utiles

Installation

Terminal

Règles

Erreurs  
courantes

Webographie

L'inclusion des modules doit se faire avant toute déclaration, et donc avant la commande `IMPLICIT NONE`.

Fortran

Serra Sylvain

Généralités

Types de données

Opérateurs

Algorithmique

Sous-programmes

Tableaux

Entrées – Sorties

Suppléments

Informations utiles

Installation

Terminal

Règles

Erreurs courantes

Webographie

## IDRIS (institut du développement et des ressources en informatique scientifique)

[http://www.idris.fr/media/formations/fortran/idris\\_fortran\\_avance\\_cours.pdf](http://www.idris.fr/media/formations/fortran/idris_fortran_avance_cours.pdf)

[http://www.idris.fr/data/cours/lang/fortran/choix\\_doc.html](http://www.idris.fr/data/cours/lang/fortran/choix_doc.html)

**Développez.com** : <http://fortran.developpez.com>

**Misfu** : <http://www.misfu.com/information-cours-tutos-tutoriaux-Fortran.html>

tous les sites universitaires et/ou persos bien faits

**ENSEIRB-MATMECA** :

[http://www.math.u-bordeaux1.fr/~lmieusse/PAGE\\_WEB/ENSEIGNEMENT/cours\\_f90.pdf](http://www.math.u-bordeaux1.fr/~lmieusse/PAGE_WEB/ENSEIGNEMENT/cours_f90.pdf)

**ENS** : <http://www.lmd.jussieu.fr/~lgldm/Enseignement/Fortran.pdf>

**Observatoire de Paris** : <http://www.obs.u-bordeaux1.fr/radio/JMHure/CoursF95CMichaut.pdf>

**Université Joseph-Fourier / INRIA** :

<http://www-ljk.imag.fr/membres/Laurence.Viry/document/f95.ps>

**V. Louvet du CNRS** :

<http://calcul.math.cnrs.fr/Documents/Ecoles/LEM2I/Mod3/fortran.pdf>

Plus tous les sites anglophones. . .