Fortran : sa Normalisation

Les organismes de Normalisation

- ISO : International Organization for Standardization
- IEC : International Electrotechnical Commission

ou CEI : Commission Électrotechnique Internationale

- IEEE : Institute of Electrical and Electronics Engineers
- ECMA : European Computer Manufacturers Association (*)

aussi

- ANSI : X3J3 (INCITS PL22.3)
- AFNOR (anciennement CN22/GE5)

(*) Exemples : ECMA-262 15th ed. 2024-06 (ex JavaScript) ; 840 pages
EMCA-334 7th ed. 2023-12 = C# ; 671 pages aussi : formats XML

Mais souvent aussi des Normes communes

Exemple : ISO/IEC 10646:2020
ISO/IEC 29500-1:2016 [office Open XML File Formats]
ISO/IEC 9899:2018 [pour le langage C]
ISO/IEC/IEEE 60559:2020 [dite IEEE 754]

ISO/IEC JTC1 (Joint Technical Committee) ← Le seul !

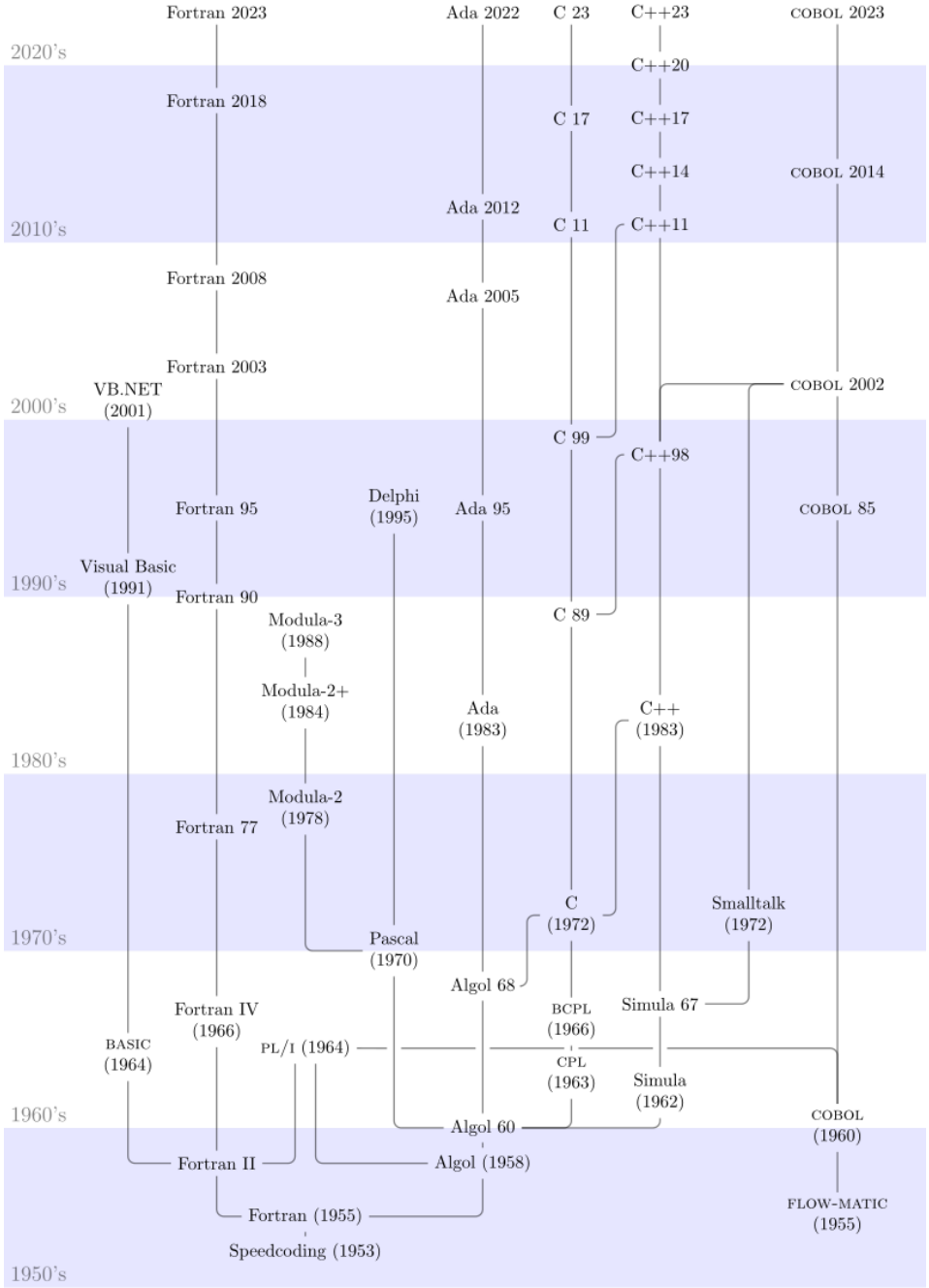
dont
SC 2 : Jeux de caractères
SC 22 : Langages de programmation
SC 25 : Interconnexion des équipements informatiques

Rmq1 : en typographie française e.g. ISO/CEI 9899 : 2018 pour le C
Rmq2 : ISO/CEI comme les systèmes temporels : UTC, TDB, TDT, TCG, TT, TAI, ..

Les groupes de travail de l’ ISO/IEC JTC1 SC22

WG4 : COBOL
WG5 : Fortran
WG9 : Ada
WG14 : C
WG17 : Prolog
WG21 : C++
WG23 : Vulnérabilités
WG24 : Linux

Les « anciens » groupes non actifs : Pascal, APL, Algol, PL/I, Basic, Modula-2, Lisp, …
Les groupes n’ayant jamais « pris » : Java Study Group



Le groupe de travail ISO/IEC JTC1 SC22 WG5 (Fortran)

Environ 30 – 40 experts :
1/2 universitaires, utilisateurs
1/2 industriels des compilateurs (Intel → Steve Lionnel , ….)

Pays les plus actifs : USA (historiquement les premiers, J. Martin,)
UK (très forte histoire : M. Ellis, M. Cohen avec NAG, M. Metcalf , J. Reid)
1 réunion formelle par an (La France a pu en organiser une en 1999 à Cadarache [merci le CEA]
sinon impossible d'y assister [merci le CNRS !!]
Mais des échanges par courriel, beaucoup de demande d'interprétation e.g.

Carlson, Neil via J3	Re: [J3] Is this code valid?	jeu. 01/05/2025 15:00
mois dernier		
Robert Corbett via J3	**Re: [J3] Is this code valid?**	**mer. 30/04/2025 04:41**
John Reid via J3	Re: [J3] Is this code valid?	mar. 29/04/2025 22:10
Daniel Chen via J3	**Re: [J3] Is this code valid?**	**mar. 29/04/2025 18:27**
Carlson, Neil via J3	**[J3] Is this code valid?**	**mar. 29/04/2025 16:36**

Après il y a formellement un vote (mais cela a été discuté par échanges de courriel …)

[exemple du vote à faire avant le 15 juin 2025 23:59 UTC]

```
The following Fortran 2023 interpretations are being balloted:
Yes  No   Number   Title
---  ---  F23/003  Conflicting rules for COMMON block names
---  ---  F23/004  OUT_OF_RANGE and ROUND argument
---  ---  F23/005  Defined assignment/operators and dynamic type
---  ---  F23/006  Underflow in IEEE_SCALB
---  ---  F23/008  Real argument I in IEEE_SCALB
---  ---  F23/009  Coarray subobject of component
---  ---  F23/010  MOVE_ALLOC with coarray arguments
---  ---  F23/011  NULL and procedure pointers
---  ---  F23/012  Coarray correspondence in DEALLOCATE
---  ---  F23/013  BOZ literals in interoperable enumerators
---  ---  F23/015  Coindexed objects in structure constructors
---  ---  F23/016  Segments associated with allocation
---  ---  F23/017  CFI_establish nonalloc nonpointer null base address
---  ---  F23/018  Correspondence of unallocated coarrays
The text of these interpretations appears below.
```

Mais aussi des propositions de nouvelles fonctionnalités

- **Une** idée et un besoin identifié : connaitre les numéros des unites d'entrée, de sortie et des erreurs par défaut (cela peut-être 5,6,7 ou 0,1,2 ou …)
- Une proposition d'ajout dans la Norme : j'avais initialement proposé trois fonctions enquêtrices ("inquiry functions") en précisant aussi que cela ne peut pas interférer avec des programmes qui existent déjà (ceux qui respectent la Norme !!)

MAIS on n'est pas tout seul et on travaille en équipe

- Dans mon cas, Van Snyder (NASA) s'y est 'collé' et c'est ce travail à 2 qui a conduit à la création du module ISO_FORTRAN_ENV puisqu'il est apparu que cela pouvait être résolu dès la compilation (et inutilement à l'exécution).

De même, la proposition d'une fonction GET_ERROR_MESSAGE pour des E/S (de portée plus limitée car non généralisable à d'autres cas) a donné lieu, après là aussi à des échanges, à l'introduction de deux nouveaux mots-clefs IOMSG= puis ERRMSG=

Avantages et inconvénients de la Normalisation

- **Respect des autres Normes en relation**

** la série des LIA [Language independent arithmetic] :

ISO/IEC 10967-1 : 2012 [Integer and floating point arithmetic] 🡐 confirmé en 2024

ISO/IEC 10967-2 : 2001 [Elementary numerical functions]

ISO/IEC 10967-3 : 2006 [Complex ….]

** ISO/IEC/IEEE 60559 : 2020 [dite IEEE 754]

** JTC1 directives [e.g. séparateur décimal « , » ; pas de « billion » ; « shall, should, may, can »

➔ Contre-exemple : Python !!!

- **Stabilité dans le temps**

Quelques caractéristiques détruites (préavis de l'ordre de 20 ans)

ou déclarées obsolètes

Contre-exemple : Python !!!

- **Obligations réglementaires** : Statut de la Normalisation en France & en Europe !!

Des langages de plus en plus « complets »
et des Normes de plus en plus « lourdes »

Langage C [ISO/IEC 9899] :

1990 → 219 pages
1999 → 538 pages
2011 → 683 pages
2018 → 520 pages
2024 → 758 pages

Langage C++ [ISO/IEC 14882] :

1998 → 732 pages
2003 → 757 pages
2011 → 1338 pages
2014 → 1358 pages
2017 → 1605 pages
2020 → 1853 pages
2024 → 2104 pages

L'histoire des Normes « Fortran »

« FORTRAN II» , FORTRAN IV , FORTRAN 66 , FORTRAN 77

Fortran 90 soit 1539 : 1991 → 369 pages

Fortran 95 soit 1539-1 : 1997 → 346 pages

Fortran 2003 soit 1539-1 : 2004 → 567 pages

Fortran 2008 soit 1539-1 : 2010 → 603 pages

➔accord du nom avec l'année !!

Fortran 2018 soit 1539-1 : 2018 → 630 pages, 1,70 kg

Fortran 2023 soit 1539-1 : 2023 → 674 pages

Fortran 2028 soit J3/25-007 Fortran 2028 WD du 2024-12-29T12:55 → 701 pages, 1,85 kg

+ Technical Report (Calcul parallèle & CoArrays, interopérabilité avec C, Exceptions IEEE, …)

Rmq : aussi donc 1539-2 : 2000 [Varying length character string] → 20 pages

et 1539-3 :1998 [Conditional compilation] ?? Disparue ??

À l'INSU, il y a eu un « Alsatran » !!

Quelques particularités de Fortran :

- Les tableaux commencent (par défaut) avec l'indice 1
  ➔ poids de l'histoire (cf. année 0, …)

- Pas de bibliothèque standard : par exemple, toutes les fonctions sont dans la Norme ← important

- Modèle objet d'Ada [France entre autres …]
  (et non celui du C++) ← excellent choix !
  [BS : Si c'était à refaire … a+2 et non 2+a]

- Révisions majeures (ajouts importants) [voir plus loin]
  1991 (F90), 2004 (F2003), ~~2018~~, 2028
  + Révisions « mineures » : 1997 (F95),
  2010 (F2008), 2018, 2023 + corrigendums

Les « grandes » nouveautés dans les Normes de Fortran

Fortran 90 : **Format libre**, **Calcul vectoriel** (le seul à ce jour), instructions WHERE,
Allocation dynamique, PUBLIC & PRIVATE, **Vocation des arguments**
Modules , **Types dérivés** , Récursivité
Fortran 95 : **PURE**, **ELEMENTAL**, MAXLOC, MINLOC, ~~FORALL~~,

Fortran 2003 : **Orienté Objet** (CLASS, ABSTRACT, FINAL, EXTENDS, SELECT TYPE,
PROTECTED, …),
Exceptions IEEE, **Interopérabilité avec le C**,
Module ISO_FORTRAN_ENV
dont INPUT_UNIT, OUTPUT_UNIT, ERROR_UNIT
Améliorations E/S (dont IOMSG= , format DT),
Interactions avec la ligne de commande, IMPORT, ENUM

Fortran 2008 : **CoArray** (codimension [], image, …) → ?? GPU ??

BP : Utilité de MPI (?) → Les compilateurs devraient faire mieux !

DO CONCURRENT , BLOCK

Des procédures et des fonctions de manipulation de bits

Quelques fonctions mathématiques (Bessel, Erf, Gamma et cie.)

Interactions avec le système d'exploitation [BP Très utile] :

execute_command_line,

compiler_options et compiler_version

Fortran 2018 : Introduction des TEAMS pour le calcul parallèle et autres modifications

Amélioration [BP Complexification ?] de l'interopérabilité C / Fortran (CFI)

Amélioration de la conformité de Fortran avec ISO/IEC/IEEE 60559 : 2011

(rmq : depuis 2020) avec des fonctions dédiées ieee_<qq_chose>

Fortran 2023 : Fonctions trigonométriques (**en degré** ‘foncd’**, demi-tour** ’foncpi’)

Test à la « C » : **(condition ? expression_vrai : expression_faux)**

typeof et classof ; « simple procedures » (plus que « pure ») ; rank

enumeration type :: …. enumerator :: …. end enumeration type

et les fonctions next, previous *[BP : VERIF & A_VOIR]*

‘AT’ comme descripteur de sortie (combine ‘A’ et la fonction TRIM)

Apparition du symbole ‘@’ : exemple

```
Real, rank(2) :: A        ! A(:,:)
….
Integer, dimension(RANG(A)) :: IND = [3, 5]
Print *, A[@IND]      ! Pour A(3,5)
```

Fortran 2028 : ??

Les différents souhaits … :

[BP] Utiles !

- ‘unsigned integer’ (difficulté d’implémentation et cohérence de la Norme …)
- Des nouvelles fonctions pour améliorer la précision des calculs, pour `x` proche de zéro :
  `log(1+x)`,`exp(x)-1`,`x-log(1+x)` ,`1/gamma(1+x)-1` et aussi (peut-être)
  `(1-cox(x))/x^2`,`(cosh(x)-1)/x^2`,`(x-sin(x))/x^3`,`(sinh(x)-x)/x^3`
  Et pour `x` proche de un : `x-1-log(x)`
- Insensibilité (optionnelle) à la casse des fonctions INDEX, SCAN, VERIFY
- Fonctions UPPER_CASE, LOWER_CASE
- Operateurs .ANDTHEN. et .ORELSE. pour résoudre les difficultés liées à l’évaluation, ou pas, de `expression_b` dans les tests `expression_a .AND. expression_b` ou `expression_a .AND. expression_b`

Les différents souhaits … :

[BP] Pourquoi pas mais attention à la complexification, difficulté de lecture …

- Fortran Template Library (FTL)
- Introduire les unités de grandeur physique : exemple

```
UNIT :: CM, INCH = 2.54 * CM
UNIT :: CM_PER_INCH = CM / INCH
UNIT :: SQINCH = INCH * INCH ! or INCH ** 2
!
REAL, PARAMETER, UNIT(CM_PER_INCH) :: CONVERT = 2.54
REAL, UNIT(SQINCH) :: A
REAL, UNIT(INCH) :: L, L2
REAL, UNIT(CM) :: C
!
A = L * L2             ! VALID -- SQINCH is compatible with INCH * INCH
C = CONVERT * L        ! VALID -- CM / INCH * INCH = CM
C = CM(L)              ! VALID -- Clearer than the previous statement
L = SQRT(A) * 5.0e-3 ! VALID -- exercise for reader
```

Les différents souhaits … :

[BP] Bof !

- Opérateurs combinés à la « C » : += , -= , *= (pb avec /=)
- « raccourci » @ [BP : à toutes les sauces] e.g. x = @ + y

Un rêve (ou presque …) :

- Comme en AdA, préciser à la déclaration d'une variable son intervalle de validité
- Aussi, intégrer un mécanisme de propagation des erreurs (type CADNA)

Avec toutes ces possibilités, que faut-il utiliser ?

- Ne pas utiliser de la syntaxe pour de la syntaxe (e.g. CLASS) lorsque ce n'est pas ce que l'on veut réellement faire ! (e.g. Allocatable suffisait …)

- Et peut-on tout connaitre (cf, l'explosion de la taille des Normes) … on n'apprend pas la POO en regardant la syntaxe (Fortran, C++, …) si on n'a pas appris les concepts d'abord et ce à quoi cela peut servir, ce que cela fait ou n'est pas utile.
  Après seulement on peut réfléchir à l'implémentation dans un langage par rapport aux performances attendues vs (par exemple) la maintenabilité.

Le mieux (à définir !!) peut-être l'ennemi de la performance

- Exemple : la récursivité _ou_ l'usage d'operateurs définis par l'utilisateur à la place de procédures (voir plus loin)

- On ne fait pas, autre exemple, de la POO pour dire que l'on en a fait (effet de mode) : exemple, vu de l'algèbre linéaire codée en POO ➔ intrinsèques, BLAS, LAPACK seront toujours plus rapides.

- Code générique (avec héritage) pour faire des calculs (hydro 1D, 2D, 3D par exemple) avec les opérateurs génériques .GRAD. , .DIV. … ou spécifiques .GRAD_1D. , .GRAD_2D. , .GRAD_3D. , .DIV_1D. , .DIV_2D., .DIV_3D.

OU mieux encore (voir plus haut) :

Subroutine GRAD (…) qui se réfère à des Subroutine spécifiques GRAD_1D, GRAD_2D ou GRAD_3D

Ce que (seul) Fortran sait faire :

- Le plus rapide (avec C, ASM, …) [et langage compilé 🡐 Qualité des compilateurs
  (aussi pour la facilité de mise au point, debug, ..)
  🡒 toujours le souci de la performance (et aussi élimination des « ralentisseurs » !)
- Calcul vectoriel
- Fonctions intégrées au langage
- Exceptions IEEE

Les compilateurs (et aussi les IDE)

Faire la différence entre les compilateurs de développement et les compilateurs de production.

Faire des tests pour qualifier les différentes options du compilateur, en particulier pour les différents degrés d'optimisation.

Rappel : en principe seuls les résultats fournis avec –O0 sont garantis par le fabriquant.