

Containers pour les traceurs et isotopes ?

Réunion “Pédalons” du 24/06/2019

David Cugnet, après discussion avec Camille Risi.

Containers pour les traceurs ?

CONTRAINTES/PARTICULARITÉS:

- Lisibilité du code / modularité → transparence de la couche “isotopes”
- Éviter les copies → pointeurs pour les variables préexistantes
- Performance → contiguité si possible
- Opérations communes père (tr) et fils (iso) → imbrication, surcharge
- Appels similaires (exemple: ISOVERIF) → factorisation dans des modules
- un (phase, mélange, radioactivité) ou quelques (chimie) traceurs pour un processus isotopique → variables locales utiles (ex eau: q[,l,s]_seri)

SOLUTION DE BASE: container non hiérarchisé (ex pour H2O)

```
TYPE trac(knd)
```

```
    INTEGER,          KIND          :: knd=REAL32

    CHARACTER(LEN=256), ALLOCATABLE :: tr_name(:)           % transported tracers names
    REAL(KIND=knd),      POINTER     :: tr_seri(:, :, :)    % transported tracers fields

    REAL(KIND=knd),     POINTER     :: q_seri(:, :)        % water vapour
    REAL(KIND=knd),     POINTER     :: ql_seri(:, :)       % liquid water
    REAL(KIND=knd),     POINTER     :: qs_seri(:, :)       % ice water

    REAL(KIND=knd),     POINTER     :: qx_seri(:, :, :, :) => NULL() % water vapour isotopes
    CHARACTER(LEN=256), ALLOCATABLE :: qx_name(:)          % " " " names

    REAL(KIND=knd),     POINTER     :: qlx_seri(:, :, :, :) => NULL() % liquid water
    CHARACTER(LEN=256), ALLOCATABLE :: qlx_name(:)         % " " " names

    REAL(KIND=knd),     POINTER     :: qsx_seri(:, :, :, :) => NULL() % ice water
    CHARACTER(LEN=256), ALLOCATABLE :: qsx_name(:)         % " " " names
```

```
END TYPE trac
```

Avantages:

- mêmes noms de variables, contiguïté.

Inconvénients:

- Possibles lourdeurs de notations
- Gestion hiérarchique laborieuse
- Extensibilité peu souple

SOLUTION PRÉFÉRABLE: container hiérarchisé

```
TYPE trac1(knd)
  INTEGER,          KIND      :: knd=REAL32
  CHARACTER(LEN=256) :: name      % tracer name
  REAL(KIND=knd), POINTER :: val(:, :) % tracer field
CONTAINS
  PROCEDURE, PRIVATE :: affect => t1_affect_r32, t1_affect_r64
  PROCEDURE, PRIVATE :: plus   => t1_plus_r32,   t1_plus_r64
  ...
END TYPE trac1

TYPE, EXTENDS(trac1) :: trac2
  TYPE(trac1), POINTER :: t2(:)          % isotope/colour
CONTAINS
  PROCEDURE, PRIVATE :: affect => t2_affect_r32, t2_affect_r64
  PROCEDURE, PRIVATE :: plus   => t2_plus_r32,   t2_plus_r64
  ...
END TYPE trac2

etc. (couleurs...)
```

Généricité de la routine (opérateurs surchargés) en fonction de l'instance de CLASS(trac1)

- un seul type de structure pour une hiérarchie traceur - isotopes - couleurs
- routines internes d'indexage (utile notamment pour la chimie), utilisées "au vol" avec des noms d'espèces explicites, ou – moins coûteux – la première fois.
- assez transparent du côté des routines principales: les opérations relatives aux isotopes sont réalisées en arrière-plan

Encapsulation des processus (chimie, fractionnement...) -> namelists. Excessif ?