Fortran について

行方大輔/牧野淳一郎

神戸大学理学研究科

惑星科学研究センター

概要

- Fortran 77 (以下, F77) ユーザを対象に、FDPS Fortran APIで使っているFortran 2003の見慣れない機能、文法について概説する。
- F77は知っていることを想定。

使っている新機能一覧

大きなもの

1. module, use

1. type (構造型、C++でいう構造体/クラス)

1. C言語と相互運用利用可能性 (組み込みモジュール iso_c_binding)

使っている新機能一覧 (続き)

細々としたもの

1. 変数宣言、サブルーチンの引数宣言の形式、値渡し、パラメータ文の形式

1. do ... end do 文

1. コメントの形式

1. 比較演算子

参考書?

Fortran 2008入門

(http://www.cutt.co.jp/book/978-4-87783-399-2.html)

- module, use についての解説あり。
- 構造体についての記述も全くないわけてばない。

module, use

モジュール宣言文法

```
module モジュール名
   [宣言部]
  [contains
      モジュール副プログラム部]
end [module [モジュール名]]
```

モジュール使用文法

```
use モジュール名
```

モシュールが定義されているソースファイルを先にコンハイルすると、何か中間形式のファイルかできる。使っているほうのコンハイルではコン ハイラがそれを参照する。

module, use (続き)

- 複数のプログラム単位で使う様々なものをまとめられる。
- パラメータ宣言、データ (common block の代わりになる)、ユーザー定義型、ユーザー定義の関数やサブルーチン等。
- 構造型はモジュール内で定義して、使うサブルーチンでモジュールを use するのが基本。
- ちなみに C/C++ には相変わらずモジュールにあたるものはない (C++には名前空間はある)。

module, use (簡単な例)

```
module sample
  integer n
  parameter (n=10)
end
program main
use sample
  write(*,*) n
end
```

コンパイル、実行:

```
% gfortran module.F90; ./a.out
```

type (derived type, 構造型)

型宣言

```
type student
  character(32) name
  integer  age
end
```

変数宣言、使用方法

```
type(student)  a
a%name="Sato"
a%age=18
```

type (derived type, 構造型)

- いわゆる構造体。
- FDPS てば、3次元へグトル型、ユーザーが定義する「粒子型」等を使用。
- フロプグラミンクズタイル云々という話もあるか、キャッシュに確実に載るようにするとかにも有用。

type (型束縛手続き)

- いつのまにか Fortran も「オフジェクト志向」に。
- 雑にいうと、ある構造型の変数を第一引数にする手続き を foo(x) の代わりに x%foo と書けるというたげ。こういうのを言語によってメッセージとかメンハー関数とかいう。
- 但し、同し名前ても別の構造体のメンハー関数なら別の関数になる。演算子も関数にてぎるので、構造体同士の演算を定義てぎる。
- 以下てば「メンハー関数(手続き)」と呼ふごとに。

type (メンバー関数の文法)

モジュールstudentmoduleの定義

```
module studentmodule
  type, public:: student
    character(32)  name
    integer age
  contains
    procedure :: print
  end type
contains
  subroutine print(self)
    class(student) self
    write(*,*) self%age
  end
end
```

メンバー関数の宣言部

メンバー関数の実装部

メイン関数の定義

```
program main
 use studentmodule
 type (student)  a
 a%name="Sato"
 a%age=18
 call a%print
end
```

Fortran てもオブシェクト志向

関数のオーバーロード、演算子のオーバーロードかできる (ヘグトル型を定義して、ヘグトル同士の加算とかする演算子を定義てぎる) (FDPS 側て提供してます)

iso_c_binding

- FortranとCの相互運用性を保証する仕掛け
- 処理系とかOS依存てばなく言語定義として公式に
- Fortran側て、ˇC側て使える変数型とか関数の宣言のしかたを用意
- 文法はなんか面倒たげど、とにかくそれに従っておけはCぐから (従ってC++からも) Fortranて宣言した構造型や関数か使える
- FDPSのFortran APIは全面的にこの仕掛けを利用

iso_c_binding (例)

```
type, public, bind(c) :: full_particle
    integer(kind=c_long_long) :: id
    real(kind=c_double)  mass !$fdps charge
    ....
    type(fdps_f64vec) :: pos !$fdps position
end type full_particle
```

- bind(c)て構造体をCからもアクセスてぎるようになる (C側てば別に同等の構造体を宣言/用意する必要あり)
- Cと互換性のあるデータ型の選択は、(kind=c_double)等とkind値を指定する。
- fdps_f64vecはFDPSて提供している倍精度3次元ヘクトル型。

細々としたこと

1. 変数宣言、サブルーチンの引数宣言の形式、値渡し、パラメータ文の形式
2. do ... end do 文
3. コメントの形式
4. 比較演算子

変数宣言

古代 (F77)

```
real a(50)
real c
parameter (c=1.0)
```

現代

```
real, dimension :: a(50)
real, parameter :: c=1.0
```

- dimension, parameter の他に色々属性をつけられる。つける時には変数名の前に ”::” を。
- 古代語てもコンパイラは文句いわない(他の新機能も基本的にそう)

do ... end do文

古代 (F77)

```
   do 10 i=1, 50
     ...
50 continue
```

現代

```
do i=1,50
  ...
end do
```

コメントの形式

古代 (F77)

```
c この行はコメントです
  x = x + 1
```

現代

```
! この行はコメントです
  x = x + 1
```

比較演算子

古代 (F77)

```
if (a .lt. b) then
   ...
end if
```

現代

```
if (a < b) then
   ...
end if
```

==, /=, <, <=, >, >= がある。

まとめ

- FDPS Fortran APIで使っているFortran 77にない機能を概説した。
- module, 構造体, iso_c_bindingが主。
- 他にも配列演算等の便利そうな機能があるが省略。