階乗のプログラムのデバッグ

プログラムの仕様

$3! := 3 \times 2 \times 1$の計算を行う。

疑問点

下記のプログラム (kaijyou_ORG.F90)で, xの要素の数が 3 だと誤った結果が表示され、要素数 1 で正しい結果が表示される理由が分からない。

kaijyou_ORG.F90 (バグがある)

```
real,dimension(3)::x
x(1)=1; x(2)=2; x(3)=3

x=1.0

do i=1,3
  x=x*x(i)
  print *,'i=',i
  print *,'x=',x
end do

end
```

```
$ ift
```

```
$ ifort kaijyou_ORG.F90 -o kaijyou_ORG.EXE
```

```
$ kaijyou_ORG.EXE
 i=           1
 x=   1.000000       1.000000       1.000000
 i=           2
 x=   1.000000       1.000000       1.000000
 i=           3
 x=   1.000000       1.000000       1.000000
```

チェック用のプログラム

kaijyou_DEBUG.F90

```
real,dimension(3)::x
x(1)=1; x(2)=2; x(3)=3

print *,'BEFORE LINE 5 x=1.0: x(1),x(2),x(3)=',x(1),x(2),x(3)
x=1.0
print *,'AFTER LINE 5 x=1.0: x(1),x(2),x(3)=',x(1),x(2),x(3)

do i=1,3
  x=x*x(i)
print *,'DO LOOP: i,x(1),x(2),x(3)=',i,x(1),x(2),x(3)
end do

end
```

```
$ ift
```

```
$ ifort kaijyou_DEBUG.F90 -o kaijyou_DEBUG.EXE
```

```
$ kaijyou_DEBUG.EXE
 BEFORE LINE 5 x=1.0: x(1),x(2),x(3)=   1.000000       2.000000
   3.000000
 AFTER LINE 5 x=1.0: x(1),x(2),x(3)=   1.000000       1.000000
   1.000000
 DO LOOP: i,x(1),x(2),x(3)=           1   1.000000       1.000000
   1.000000
 DO LOOP: i,x(1),x(2),x(3)=           2   1.000000       1.000000
   1.000000
 DO LOOP: i,x(1),x(2),x(3)=           3   1.000000       1.000000
   1.000000
```

5行目の

```
x=1
```

によって, **xのすべての要素**に**同じ値** (1.0)が**代入されている**ことに注意。

デバッグのコツ

なるべく細かく数値を書き出して動作を確認する

元のプログラム (kaijyou_ORG.F90)でも下記のように, 変数の値を書き出してチェックしているのは大変良い試みである。

```
  print *,'i=',i
  print *,'x=',x
```

さらに上達するためのコツとしては, 次を覚えておくとよい

動作が不明な箇所の直前から変数の値の変化を調べる

```
print *,'BEFORE LINE 5 x=1.0: x(1),x(2),x(3)=',x(1),x(2),x(3)
x=1.0
print *,'AFTER LINE 5 x=1.0: x(1),x(2),x(3)=',x(1),x(2),x(3)
```

ここでは, `x=1.0`の直前と直後でxの値を比較している。

動作が分かりにくい箇所の変数の値を詳しく見る

```
do i=1,3
  x=x*x(i)
print *,'DO LOOP: i,x(1),x(2),x(3)=',i,x(1),x(2),x(3)
end do
```

doループの中で,

- xの値が毎回書き換えられる
- xが配列のため、各要素の値が分かりにくい

ため,

- ループの中で毎回xの値を書き出す
- 配列xの要素をすべて書き出す

という工夫をしている。

配列の要素数が多い場合, いくつか代表的な要素だけ書き出してみる (例：最初と最後, 中央など)。

プログラムの修正

計算方法(アルゴリズム)

$$3! = 3 \times 2 \times 1 = 1 \times 2 \times 3$$

と考えて,

```
FACT=FACT*x(i-1)
```

を`i=1`から`i=3`まで繰り返す。

修正したプログラム

kaijyou_FIXED.F90

```
real,dimension(3)::x
x(1)=1; x(2)=2; x(3)=3

i=1;FACT=x(1)
print *,'i,x(i),FACT=',i,x(i),FACT

do i=2,3
  print *,'BEFORE i,x(i),FACT=',i,x(i),FACT
  FACT=FACT*x(i)
  print *,'AFTER i,x(i),FACT=',i,x(i),FACT
end do

print *
print *,'RESULT: FACT=',FACT

end
```

実行例

```
$  kaijyou_FIXED.EXE
 i,x(i),FACT=           1   1.000000       1.000000
 BEFORE i,x(i),FACT=           2   2.000000       1.000000
 AFTER i,x(i),FACT=           2   2.000000       2.000000
 BEFORE i,x(i),FACT=           3   3.000000       2.000000
 AFTER i,x(i),FACT=           3   3.000000       6.000000

 RESULT: FACT=   6.000000
```

文法のポイント

Fortranの配列の代入ルール

例えば,

```
REAL,DIMENSION::X(3)
```

という配列Xに対して,

```
X=1.0
```

とすると, 以下のように処理される。

```
X(1)=1.0
X(2)=1.0
X(3)=1.0
```

Xのすべての要素に同じ値 (1.0)が代入される。