覚書；Fortran 90/95 によるプログラミング

1. なぜ Fortran なのか．今更，今なお，……　　餅は餅屋

   > FORTRAN is well suited to numerical computations, C is suitable for system programming tasks. A combination of the two languages in one program should have been the best solution for some programs.
   > — USER NOTES ON FORTRAN PROGRAMMING (UNFP)

2. コーディングでは三つのことを守ればよい．あとは些細なことで，経験を重ねれば落ち着く所に落ち着く．　　蓼食う虫も好き好き

   - 手続き，関数には簡潔なコメント ヘッダをつける．参照した方程式，アルゴリズムがあればその出所を（ヘッダに）示す — 出所だけ記す．
   - 適当な字下げをする．（使うのであれば）タブは空白で埋める．
   - 変数が表す値の単位を明示する．多くの場合変数のコメントは単位だけでよい．

3. “Internal Sub-Program” は使用しない．“Module" を利用する．　　適材適所

4. 継続行の行頭は +, -などの二項演算子がくるようにする．数式の慣例に従う（個人的には行末に演算子がくる方が読み易い）．　　藪を突突いて蛇を出す

   > 式が長くて途中で行がかわるときには，新しい行の頭に ×, +, − をつける（行末と，新しい行の頭と，両方につける流儀もある）．— 木下是雄著，『理科系の作文技術』

   「行末に書いてあるだけでは行が長いと見逃してしまう．そこで，行の先頭にも ‘&’ を書くこと」とする意見もある．FORTRAN 77 であるまいに，行末だけにする．

5. 単語の区切りに大文字を使う Pascal 風な命名 (InfixCaps) はしない．（数学）記号の大文字，小文字の区別と紛らわしい．
   単語の区切りに ‘_’ を使うか使わないかはサブ プログラム単位で統一する — FORTRAN 77 のコード作成・保守時の注意．　　木に竹を接ぐ

6. 略語はプログラムで統一する．略語表をソースと共にすればなお好い．　　正直の頭に神宿る

   黒子には黒子らしい名前をつける．重要でない変数の名前は凡庸な目立たない名前とする．
   e.g. `t1 (tmp_1/temporary_1st)`

   英語圏のプログラマでも四字単語やハイフネーションに気を使っていないようだ（発音のしやすさに重きを置いているようだ）．　　木を見て森を見ず

7. ‘*’, ‘/’, ‘**’ の前後には空白を入れる．配列の添え字式等の空白は眺めて決める．カンマの後ろには必ず入れる．　　枝を撓めて花を散らす

```
y = a*x**2 + b*x**2 + c          ! usual
y = a * x ** 2 + b * x ** 2 + c ! I prefer

a(i), a(i, j), a(i+1) or a(i + 1), a(i+1, j+1)
s(1:6), s((i+1):(i+6))

* Blank characters and blank lines should be used freely in
  Fortran programs. This improves readability.
* Blanks should be placed before and after the percent sign
  in structure component references. This is not required,
  but it improves readability.
--- Fortran 90 programming tips by Walt Brainerd
```

空白はその前後・左右の関連性を希薄にし読み難くすることもある.

8. 'SAVE' は使うな．理由は多々あるが一つには DLL 内の "save" 属性の変数は意図したように初期化されない（コンパイラに依存するかもしれない）.
   Antony J Mee, "Mistakes in Fortran 90 Programs That Might Surprise You" を読むこと.

君子危うきに近寄らず

9. 二段字下げはしない．Pascal (/Basic) ではない.

郷に入っては郷に従う

```
select case (wrong)     select case (good)     if (1 == good) then
  case (1)              case (1)                   ...
      ...                   ...                else if ( ...
  case default          case default           else
      ...                   ...                    ! relax
end select              end select             end if
```

10. 文脈で内容，役割が明確であるなら識別子の長さは短いほど判り易く読み易い.

多弁能無し

```
! Convert pressure P in unit U into Pc in MPa
function find_pressure_in_MPa(P, U) result(Pc)
```
U 単位の圧力 P を MPa 単位に換算した圧力 Pc を返す.
```
function find_P_MPa(P, U) result(Pc)
```
MPa は単位であるのが明白なので in は冗長である.
```
function P_MPa(P, U) result(Pc)
```
手続き名，"IS" 関数でない関数名は動詞で始めるべきだ.

```
Energy = mass * (speed_of_light ** 2)
```
他の言語ならともかく Fortran では `E = m * (c ** 2)`

単語の区切りに '_' を使う場合で，特に紛らわしくない時には前置詞は省く.
('_' が '␣', ' of ', ' in ', ' per ' 等かは文脈で判る.)

11. '(', ')' とうち，← キーを押すように，subroutine とうったら implicit none とタイプする.
    「(変数名の）タイプ ミスを防止するため」だけに使うのであれば書くな.

加藤一二三の棒銀

12. 文字定数の両端を区切る文字はアポストロフィとする．二重引用符は用いない．‘string’ 型は（まだ）ない．

13. Rob Pike, “Notes on Programming in C”, 1989 を 読め． 温故知新
英文（スタイル）に注目しよう．

14. 一手間かける．利用できるのであれば “Static Analyser” を使おう． 傍目八目

戯言；コーディング

1. Delphi でチェック ボックスの名前にわざわざ ‘chk’ 接頭辞をつけるのに，‘chkIsTaxAdded’ とするような愚はおかすな．‘chkTaxAdded (/chkTaxAdd)’ でよい．

2. 文脈によっては「ハイフネーション」より「分綴」の方がよいことがある．たとえ，「ぶんてつ」と読めなくても（特に日本人の私には）判り易いことがある．欧文でも頭字語がある，長いと彼女・彼も読みにくいのだ．

   プログラムを音読することは全くといっていいほどない．プログラムでは音読し易さより識別し易さを優先したがよい．

3. 不揃の調和・美もある．不揃の調和・美を受けいれるにはそれなりの力がいる．判らないのは想像力の欠如による．

4. 「スコープは小さく・狭く」といいながら，132 カラム行を使うのはおかしくはないか．
   — 私はおかしいと思う．A4 版横置き一段組みの本を私は見たことがない．