# これはなに

皆さんはこれまでの課題でFortranのコードがだいたい読めるようになってきたと思います。読めたら、次は**書く**番です

ということで、Fortranのコードを書く際に、必要な基礎知識を紹介していきます。僕は文章書くのが苦手なので読みづらいと思います。頑張って読んでください。

# 自由形式と固定形式

Fortranの書き方は大きく2種類に分かれています。自由形式と固定形式です。ネット上などでは、固定形式で書かれたFortranを「FORTRAN」、自由形式で書かれたFortranを「Fortran」と書き分けることが多いです。この二つの形式の違いをかんたんにリストアップすると、

固定形式

- 行頭に6個分空白を開ける
- 暗黙の型宣言が有効
- common文が使われる
- 行番号が多く使われる
- goto文が使われる

自由形式

- 行頭に空白は不必要(可読性のために空白を入れることはある)
- 暗黙の型宣言は無効
- common文、行番号、gotoは使わない。

自由形式のほうが新しい書き方で、推奨されています。固定形式はバグを生みやすい書き方になりやすく、推奨されていません。

特にcommon文、行番号、goto文はコードがぐちゃぐちゃになるため、新しくコードを書く際は極力使わないほうが良いです。

今までみなさんが課題で見てきたコードは固定形式で書かれたコードでした。この研究室には固定形式で書かれたコードが数多く眠っており、読み方くらいは知っておかないと困るので、今までの課題は固定形式のコードを用いて行っていました。

これからの課題では、なるべく自由形式のプログラムを用いることにします。みなさんもコードを書くときは自由形式で書くようにしてください。

# 自由形式基本のキ

ここでは自由形式を書く際の基本について説明していきます。「変数」や「宣言」などの、固定形式でも出てきていた概念にはあまり触れません。

自由形式の超超基本的な書き方は以下のとおりです。

```
program main
    ↑
    ここの間にプログラムを書いて行く。
    ↓
end program
```

`program main ~ end program` の間はメインルーチンと言われています。コードをコンパイルして実行すると、メインルーチンのコードが上から一行ずつ処理されていきます。`main` というのはメインルーチンの名称で、好きな名前にしてもらって良いです。

もうちょっと具体的な話をすると、自由形式のコードは以下の構成になっていることが多いです。

```
program main
    implicit none           ←暗黙の型宣言の無効
    <使用する変数の宣言>      ←コードより必ず上
    <コード>                 ←宣言より必ず下
end program
```

要点だけまとめると、「一番上に `implicit none` を書き、使う変数を宣言して、その下の行からコードを書き始める」という感じです。コードの合間に変数の宣言を挟むことは基本出来ません。必ずコードの書き始め位置よりも上に変数の宣言を書くようにしてください。

次に細かい説明をしていきます。上の要点だけ抑えてくれればコードは書けるので、読み飛ばしてもらっても構いません。

`implicit none` は暗黙の型宣言の無効と言われています。そもそもFortranには、「a-h, o-zから始まる名前の変数は実数」、「i-nから始まる名前の変数は整数」、と変数の型が自動的に設定される便利機能があります。これを**暗黙の型宣言(implicit)**と呼びます。`implicit none` はこの設定を**無効(none)**にしてください。という意味になっています。

つまり、自由形式では「 implicit none によって、コード上で用いる変数はすべて宣言する必要がある。」ということです。どうして暗黙の型宣言という便利機能をわざわざ無効にするのでしょうか。以下のコードを見てみましょう。

```
program example
    integer(4):: longlongnamevariable

    longlongnamevariable=10
    longlongnamevariable=longlongvariablename+5

    print*, longnamevariable
end program
```

このプログラムの動きは、最初に longlongvariablename という名前の変数に10を代入します。次に自分自身を+5します。最後に出力をします。15という値が出力されるはずです。しかし、このプログラムは2箇所バグがあるため、正常には動きません。

自分自身に+5する際の変数名と、出力する際の変数名にスペルミスがあります。これがバグです。明らかにバグを含むプログラムは、コンパイル時にエラーが出るため実行前に気づくことができますが、このプログラムはコンパイルが通り、実行時にもエラーは出ることなく通常終了します。これは暗黙の型宣言が有効になっているためです。

しかし暗黙の型宣言を無効にしてやれば、コンパイル時にエラーが出てくれるため、デバッグ(プログラムのバグを取る)の効率化につながるというわけです。

# さて

文章を読んでいるだけでは、プログラミングは上達しません!「書く→コンパイル→実行→バグる→デバッグ→実行→バグる→デバッグ→実行→以下無限ループ」というルーティーンで人の心は成長します。ということで、そのための課題をいくつか課します。

課題中に出てくるコードは、手元のPCで実行してください。理解の手助けになります。「こんなコード実行するまでもねえぜ」と感じたとしても実行してみてください。コピペでも良いので。