計算物理学II (第５回)

今回の内容

- 数値計算プログラミング
- FortranまたはCの基本的な書き方
- コンパイル・実行
- 四則演算・do/for文・配列・組み込み関数

計算物理

- 物理の問題の多くは微分方程式に帰着
- 解析解はないものがほとんど
- 数値的な近似解を求める
- 計算物理は理論物理、実験物理に並ぶ第３の手法

数値計算のプログラミング言語

- Fortran (Formula Translation)
  - 数値計算に特化
  - 歴史がある(1950年代~)
  - FORTRAN66/FORTRAN77/**Fortran90/95**/Fortran2003/Fortran2008
- C言語・C++
  - 歴史がある(1970年代~)
  - 汎用
- Python
  - 新しい(1990年代~)
  - 汎用、テキスト処理・機械学習などの豊富なライブラリ
- Julia
- 初めての人は同時並行で複数言語に取り組まず、1つだけ選ぶ方がよい。
  - 2つ目の言語は1つ目よりずっと簡単に習得できる

Fortran/Cによるプログラミングの流れ

コンパイラ型言語：コンパイル作業が必要

- エディタ(emacs)でプログラムソースコードを書く
  ソースコード：人間が読める形式(Fortran/C)

- コンパイルして実行ファイルを作成
  実行ファイル：コンピュータが理解できる形式

- シェルで実行ファイル(プログラム)を実行。計算結果を出力

Fortranのサンプルコード

まずは簡単なコードを作って実行してみよう。emacsで以下のexample.f90ファイルを作る。

```
hinohara.nobuo.ga@icho:~/compphys2/fortran$ emacs example.f90 &
```

ファイルの内容は

```
program example
  implicit none
  real*8 :: a, b
  a = 1.0d0
  b = 2.0d0
  write(*,*) "sum of a and b = ", a+b
  write(*,*) "difference of a and b = ", a-b
  write(*,*) "product of a and b = ", a*b
  write(*,*) "quotient of a and b = ", a/b
end program example
```

ヒント：

ホームディレクトリで行わず作業用のディレクトリを作ってその中で作業しましょう。
サテライト室でemacsを実行するときは最後に&をつけるとバックグラウンドで実行されるため
emacs起動中も端末が使えます。
emacsで開くファイル名の拡張子をf90としておくとemacs側でFortranソースコードと認識する。

・自動的に色付け
・タブキーを押すと段下げを自動で行う
・最後のend programは endと打ってタブキーを押すと補完。
・スペースの有無は気にしなくてよいです(implicit noneの間などにはスペースが必要)

Fortranコードのコンパイルと実行

Fortranのコンパイルはgfortranコマンドで行う(**コンパイルに成功した場合は何も表示されない**)
gfortran ソースファイル名 -o 実行ファイル名　：-o 実行ファイル名は省略してもよい。
実行ファイル名を省略するとa.outという名前の実行ファイルが作成される。

```
hinohara.nobuo.ga@icho:~/compphys2/fortran$ ls
example.f90  example.f90~
hinohara.nobuo.ga@icho:~/compphys2/fortran$ gfortran example.f90
hinohara.nobuo.ga@icho:~/compphys2/fortran$ ls
a.out*  example.f90  example.f90~
hinohara.nobuo.ga@icho:~/compphys2/fortran$ ls -l
合計 20
-rwxr-xr-x 1 hinohara.nobuo.ga 9056 10月  28 16:41 a.out*
-rw-r--r-- 1 hinohara.nobuo.ga  266 10月  28 16:41 example.f90
-rw-r--r-- 1 hinohara.nobuo.ga  266 10月  28 16:27 example.f90~
hinohara.nobuo.ga@icho:~/compphys2/fortran$ ./a.out
 sum of a and b =    3.0000000000000000
 difference of a and b =   -1.0000000000000000
 product of a and b =    2.0000000000000000
 quotient of a and b =   0.5000000000000000
hinohara.nobuo.ga@icho:~/compphys2/fortran$
```

実行ファイルが作成される

実行可能(x)となっていることがわかる

実行時は./をつける

プログラムの実行結果が表示される。

作成された実行ファイルの実行は a.out ではなく **./a.out** とする
(a.outだけでは同名のコマンドを探しに行ってしまうためカレントディレクトリにあるa.outファイルであることを明示的に書く)

C言語のサンプルコード

まずは簡単なコードを作って実行してみよう。emacsで以下のexample.cファイルを作る。

```
hinohara.nobuo.ga@icho:~/compphys2/c$ emacs example.c &
```

ファイルの内容は

```c
#include<stdio.h>

int main(){

  double a,b;

  a = 1.0;
  b = 1.0;

  printf("sum of a and b = %lf\n", a+b);
  printf("difference of a and b = %lf\n", a-b);
  printf("product of a and b = %lf\n", a*b);
  printf("quotient of a and b = %lf\n", a/b);

  return 0;

}
```

ヒント：

ホームディレクトリで行わず作業用のディレクトリを作ってその中で作業しましょう。
サテライト室でemacsを実行するときは最後に&をつけるとバックグラウンドで実行されるためemacs起動中も端末が使えます。
emacsで開くファイル名の拡張子をcとしておくとemacs側でC言語のソースコードと認識する。
・自動的に色付け
・タブキーを押すと段差げを自動で行う

C言語コードのコンパイルと実行

C言語のコンパイルはgccコマンドで行う(**コンパイルに成功した場合は何も表示されない**)
gcc ソースファイル名 -o 実行ファイル名　：-o 実行ファイル名は省略してもよい。
実行ファイル名を省略するとa.outという名前の実行ファイルが作成される。

```
hinohara.nobuo.ga@icho:~/compphys2/c$ ls
example.c  example.c~
hinohara.nobuo.ga@icho:~/compphys2/c$ gcc example.c
hinohara.nobuo.ga@icho:~/compphys2/c$ ls
a.out*  example.c  example.c~
hinohara.nobuo.ga@icho:~/compphys2/c$ ls -l
合計 16
-rwxr-xr-x 1 hinohara.nobuo.ga 8608 10月 28 16:33 a.out*
-rw-r--r-- 1 hinohara.nobuo.ga  271 10月 28 16:33 example.c
-rw-r--r-- 1 hinohara.nobuo.ga   56 10月 28 16:32 example.c~
hinohara.nobuo.ga@icho:~/compphys2/c$ ./a.out
sum of a and b = 2.000000
difference of a and b = 0.000000
product of a and b = 1.000000
quotient of a and b = 1.000000
hinohara.nobuo.ga@icho:~/compphys2/c$
```

実行ファイルが作成される

実行可能(x)となっていることがわかる

実行時は./をつける

プログラムの実行結果が表示される。

作成された実行ファイルの実行は a.out ではなく **./a.out** とする
(a.outだけでは同名のコマンドを探しに行ってしまうためカレントディレクトリにあるa.outファイルであることを明示的に書く)

コンパイルエラー

- コンパイルに失敗した場合はエラーメッセージが出力されて実行ファイルが生成されない。
- プログラム言語の文法と違うことを書いていると起きる。

```
hinohara.nobuo.ga@icho:~/compphys2/fortran$ gfortran example.f90
example.f90:5:2:

   b = 2.0s0
  1
Error: Unclassifiable statement at (1)
```

Fortranのサンプルコードでdと書くべきところを
間違えてsとすると
example.f90の5行目は分類不能な文章であるという
エラーメッセージが出る。

```
hinohara.nobuo.ga@icho:~/compphys2/c$ gcc example.c
example.c: In function ‘main’:
example.c:10:3: error: expected ‘;’ before ‘printf’
   printf("difference of a and b = %lf\n", a-b);
   ^
```

C言語のサンプルコードで文の終わりの;を削除した場合
example.cの10行目で”;” がprintfの手前にあるはず、という
エラーメッセージが出る。

エラーメッセージは問題点を探す手がかりになるので英語だが読むこと
エラーの原因がわからない場合は気軽に質問してください

演習

- 変数・代入文
  - プログラムでは数値はメモリ上の領域(変数)に保持し、必要に応じて四則演算などを行う。変数への値や計算結果の代入には代入文を使う。
- 四則演算
  - サンプルコードの通り。
- doループ(C言語ではforループ)
  - プログラムの基本は上から下に順番に実行するがdoループを使うと特定箇所を繰り返して実行
- 配列
  - 変数を複数並べたもの。ベクトルや行列を表現
- 組み込み関数
  - 三角関数、指数関数などを計算する組み込み関数が用意されている

演習の進め方

- Fortran・C言語の文法まとめ
  - 先に全てを読む必要はないので演習問題と並行して必要な部分だけを参照

- 演習問題
  - サンプルプログラムはwgetコマンドでLinuxに直接ダウンロード
  - tarコマンドで解凍
  - サンプルをまずはコンパイルして実行
  - サンプルプログラムを見ながらプログラムの動作を理解
  - その後写しても構わないので自分で同じものを書いてみる
  - まずプログラムに慣れることが重要