

## 2022年度前期 情報処理演習

2022/04/20

平山 修久 (減災連携研究センター)



名古屋大学減災連携研究センター  
Disaster Mitigation Research Center, NAGOYA UNIVERSITY

## 履修上の注意と質問への対応

### › 履修上の注意

- ー 学生同士で内容をコピーした場合には、コピーさせた学生とコピーした学生の両方とも採点しない。

### › 評価方法

- ー 毎回の演習レポートと総合演習レポート
- ー ただし、提出回数が8回未満の場合は評価しない。

### › 質問など

- ー Zoomチャット、Email等。
- ー 講義資料や課題解答は、NUCT、Web (<https://hirayamalab.com/lecture/>) で。

## 本講義の概要

- › 担当：平山 修久 (減災連携研究センター)  
減災館3階306号室  
Email : [hirayama.nagahisa@nagoya-u.jp](mailto:hirayama.nagahisa@nagoya-u.jp)
- › 講義時間：前期・水曜3限 (13:00~14:30)
- › 教室：Zoom
- › 演習の進め方
  - ー 講義と演習。演習の課題は授業時間内に終了できる程度の内容。
  - ー 演習レポートの提出は、NUCT

## IB北10階サテラボ講義室の使用ルール (念のため)

- › 通常の利用時間：8:30~18:00

- ー 土日祝日は利用不可

- › ゴミを放置しない。持ち帰る。

- › ホワイトボード使用後は消す。

- › 使用後は、照明や空調を切ること。机のフタも元に戻す。

- › 飲食厳禁。

## 本演習の目標

1. プログラムの**アルゴリズム**を理解する
2. Fortranの文法を理解し、プログラムの読み書きができる
3. 与えられた課題を解くためのアルゴリズムを考えることができる
4. Fortranによる**コンピューターでの数値解析**（方程式、モンテカルロ法）ができる

## 本講義をはじめるにあたって

- › コンピューターで計算・解析を行う。
- › 汎用プログラミング言語を使ってプログラムを作し、実行する。
  - Fortran, C言語, Mathematica, MAPLE, R, pythonなど
- › 本講義では、「Fortran」

## 情報処理

1. 情報処理序説（山本俊行先生、必修 1年春学期）  
計算機、プログラム
2. **情報処理演習**  
本講義
3. 数値解析学（中井健太郎先生、選択、2年秋学期）  
環境土木工学で用いる微分方程式の数値解析

## 本日の目標

- Fortranを実行できる環境を整備する。

## Windowsの場合

1. gfortran - the GNU Fortran complier, part of GCCのwebsiteにアクセスし, DownloadからBinaries for Windows, Linux, MacOS and much more!のBinariesをクリックする。
2. TDM GCCをクリックし, TDM GCCのwebsiteからgfortranをインストールする。tdm64-gcc-9.2.0.exeを保存し, 実行する。インストール時に, 「New Installation:Choose Components」の時に, gcc/fortran に必ずチェックをつけてインストールする。

## macOSの場合

1. まず, Xcodeをインストールする。もし, macOSのバージョンにより最新のXcodeがインストールできない場合には, Apple Developerにサインインすることで, macOSのバージョンに合うXcodeを見つけることができる。
2. command line toolsをインストールする。
3. gfortran - the GNU Fortran complier, part of GCCのwebsiteにアクセスし, DownloadからBinaries for Windows, Linux, MacOS and much more!のBinariesをクリックする。
4. macOSにある<https://github.com/fxcoudert/gfortran-for-macOS/releases>から自分のmacOSのバージョンのdmgファイルをダウンロードし, インストールする。

## インストールできない場合

- ideone.comでfortranが実行できます。本講義の課題は, ideoneで実行できることを確認しています。

## 本日の目標

- › (fortranを実行できる環境を整備する。)
- › プログラムを実行できる。
- › レポートをNUCT上で提出できる。

## プログラム作成, 実行の流れ

1. プログラムを書く
2. 書いたプログラムを動くようにする
3. プログラムを実行する

### Step 1. プログラムを書く

- › エディタ（テキスト文章を作成, 編集するアプリケーション, Windowsではメモ帳, Macではテキストエディット, UNIX: viなど）を使って, キーボードでプログラムを打ち込み, ファイルに保存。
- › ファイルの内容（ソースコード）は人は理解できるが, CPUは理解できない。

### Step 2. プログラムを動くようにする

- › コンパイル
- › ソースコードを機械語に変換（コンパイル）する。
- › Fortranであれば, Fortranコンパイラ
- › プログラムに間違いがなければコンパイルに成功。実行ファイル (a.exe, a.out) が作られる。
- › 書いたプログラムの文法に誤りがあればコンパイルに失敗し, コンパイルはエラーを表示。その場合には, ソースコードを修正し, 再度コンパイル。

### Step.3 プログラムを実行する

- › コンパイルで作成された実行ファイルを動かす。
- › OSのコマンドライン（Windows: コマンドプロンプト, Mac: ターミナル）で, 実行ファイルのファイル名を入力。

## プログラムの実行 (Windows)

### > Step 1 プログラムを書く

- エディタ（メモ帳）でソースコードを入力し、ファイル「\*.f90」（拡張子）で保存。

### > Step 2 プログラムをコンパイルする

- コマンドプロンプトで、「gfortran プログラム名」で、「a.exe」の実行ファイルを生成。

### > Step 3 プログラムを実行

- コマンドプロンプトで、「a.exe」と入力してEnter

## コマンドプロンプトの基本 (Windows)

### ターミナルの基本 (macOS)

「>」マークの左側は現在のディレクトリ（カレントディレクトリ）。

基本的なコマンド：

`cd 書式：cd パス名`

パス名を省略すると、カレントディレクトリのパスを出力。Zドライブの直下にある「fortran」ディレクトリに移動したら、「z:¥fortran>」となる。

`dir (macOSは ls) 書式：dir パス名`

指定したパスのディレクトリやファイル一覧を表示。パスを省略すればカレントディレクトリの情報を表示

`cls (macOSは clear)`

コマンドプロンプトの表示をクリアする。

`exit`

コマンドプロンプトを終了する。

「↑」 「↓」

入力履歴の表示

## 実際にプログラムを動かしてみよう1/4

- エディタを開く。スタート→プログラム→アクセサリ→メモ帳、もしくはWindows+R→notepad.exeを入力
  - 配布した資料のプログラム例を入力。入力は全て半角。プログラムを見やすくするため、字下げするとよい。
  - 書いたプログラムを、「sample0421.f90」という名前のファイルとして、fortranディレクトリ内に保存する。
  - ファイル名が「sample0421.f90.txt」となった場合は、「.txt」を削除。
- > macOSの場合
- 「テキストエディット」を起動。
  - ファイル名は、最後に「.f90」をつける。

## 実際にプログラムを動かしてみよう2/4

- コマンドプロンプトを起動。デスクトップ上のショートカットをクリック、スタート→プログラム→アクセサリ→コマンドプロンプト、Windows+R→cmdと入力
  - プログラムファイルを保存したディレクトリ「fortran」に移動
- ```
cd ¥fortran
macOS:
```
- > コンパイラgfortranを使ってコンパイルする。
- ```
gfortran sample0421.f90
```
- > コンパイラに成功するとプログラムファイルが存在するディレクトリ内にa.exe、macOSではa.outという実行ファイルが作成される。

## 実際にプログラムを動かしてみよう3/4

- > 実行ファイルを実行する。コマンドプロンプト上で `a.exe` と入力して Enter

`a.exe`

- > macOS : 実行ファイルを実行する。ターミナル上で `./a.out` と入力して Enter

`./a.out`

## ソースコードを修正する4/4

- > 【Windows】 `.f90` の拡張子のファイルを開く
  - 「`.f90`」の拡張子のファイルをダブルクリック
  - ウィンドウが出現したら、 **一覧からプログラムを選択する** にチェックを入れて OK をクリック
  - 「**NotePad (メモ帳)**」をクリック
  - この種類のファイルを開くときは、 **選択したプログラムをいつも使う** にチェックを入れて OK をクリック
- > 【macOS】 `.f90` のファイルを右クリックし、 テキストエディットで開く

## スクリーンショットを撮る

- > 【Windows】
  - キーボードの「PrintScreen」、 「Alt + PrintScreen」キーを利用する。
  - PrintScreen でデスクトップ全体
  - Alt + PrintScreen で最前面のウィンドウのみ
  - その後、ペイント（「スタート」→「プログラム」→「アクセサリ」→「ペイント」）を起動し、貼り付け（Control + V、もしくは右クリックで貼り付け）
  - ファイルを保存
- > 【macOS】
  - ⌘+Shift+3、もしくは ⌘+Shift+4

## データ型、定数、変数

- > Fortran には 5 種類の基本データ型
  - INTEGER : 整数 0, 137, -2516, 17745
  - REAL : 実数 1.234, -0.01536, 56473., 3.37456E2
  - COMPLEX : 複素数 (a, b), (1.0, 1.0), (-6.0, 7.2)
  - CHARACTER : 文字列、二重引用符またはアポストロフィで囲まなければならず、両端に同じ記号。
  - LOGICAL : 論理変数, .TRUE., .FALSE.

# IMPLICIT NONE文

## プログラムではつけること。

- › Fortranでは、型宣言文で明示的に型を宣言しない変数は、**暗黙の型宣言**に従って型を割り当てられる。名前がI, J, K, L, M, Nおよび対応する小文字で始まる未宣言の識別子は整数型 (INTEGER) となり、それ以外はすべて実数型 (REAL)
- › この暗黙の型宣言を無効にする文

IMPLICIT NONE

# 入出力

- › PRINT \*, 出力並び

```
PRINT *, "At time", Time, "seconds"
PRINT *
```

- › READ \*, 入力並び

```
READ *, InitialHeight, InitialVelocity,
Time
100.0, 90.0, 4.5
100.0 90.0 4.5
100.0 90.0
4.5
```

# 入出力 (IDEONE)

The screenshot shows the ideone.com interface. The code editor contains the following Fortran program:

```
1 program TEST
2     read *, a, b, c
3     Print *, "a:",a, "b:",b,"c:", c
4 end
```

The output window shows the following error message:

エラーが出て  
プログラムを ————— プログラムファイル  
修正する ————— をダウンロードする

修正するボタンが赤枠で囲まれています。

下方の標準入力 (stdin) ボックスには「15」と入力されています。このボックスも赤枠で囲まれています。

下方の標準出力 (stdout) ボックスには「Standard input is empty」と表示されています。

# プログラムの構成と書式

- › Fortranプログラムの一般的な構成

- PROGRAM kadai
- 宣言部
- 実行部
- 副プログラム部
- END PROGRAM kadai

## データ型, 定数, 変数

› Fortranには5種類の基本データ型

**INTEGER** : 整数 0, 137, -2516, 17745

**REAL** : 実数 1.234, -0.01536, +56473.,  
337.456, 3.37456E2

**COMPLEX** : 複素数 (a, b), (1.0, 1.0), (-6.0,  
7.2)

**CHARACTER** : 文字列, 二重引用符またはアポスト  
ロフィで囲まなければならず, 両端に同じ記号。

**LOGICAL** : 論理変数, .TRUE., .FALSE.

## 識別子

› プログラム, 定数, 変数など, プログラム中の要  
素を識別するために使う名前。

› 英文字で始まり, 最大30文字までの英文字, 数  
字, 下線

Mass

Rate

Velocity

Speed\_of\_Light

— Fortran90では大文字と小文字を区別しない。

— 一般的に, Fortranのキーワードは大文字, プログラ  
マが定義する識別子は小文字 (最初も文字は大文字)

## 変数

› Fortran変数の型は, その変数に代入される値の  
型を決定する。ゆえに, 型宣言文によって,  
Fortranプログラムの各変数の型を宣言する。

INTEGER :: リスト

REAL :: リスト

INTEGER :: NumValues, Sum

REAL :: Mass, Velocity

CHARACTER(n) :: リスト

CHARACTER(15) :: FirstName, LastName

CHARACTER :: Initial

CHARACTER(15) :: Name, L\_Name\*20, Initial\*1

## IMPLICIT NONE文 プログラムではつけること。

› Fortranでは, 型宣言文で明示的に型を宣言しない  
変数は, 暗黙の型宣言に従って型を割り当てられ  
る。名前がI, J, K, L, M, Nおよび対応する小文  
字で始まる未宣言の識別子は整数型 (INTEGER)  
となり, それ以外はすべて実数型 (REAL)

› この暗黙の型宣言を無効にする文

IMPLICIT NONE

## 変数の初期化

> Fortranではすべての変数が初期状態では未定義

```
REAL :: W = 1.0, X = 2.5, Y = -7.73
```

> 名前付定数：PARAMETER属性

型指定子, PARAMETER :: リスト

```
INTEGER, PARAMETER :: Limit = 50
```

```
REAL, PARAMETER :: Pi = 3.141593, TwoPi =  
2.0 * Pi
```

```
CHARACTER(*), PARAMETER :: Units = "cm"
```

## 今日の課題

1. 「kadai0420\_1.f90」を実行する。提出物：実行結果のスクリーンショット
2. PRINT文を用いて、自分の学籍番号と名前（ローマ字）を表示するプログラムを作成、実行する。提出物：プログラムファイル
3. 以下の計算式をPRINT文と用いて実行し、表示される答えの違いを知る。

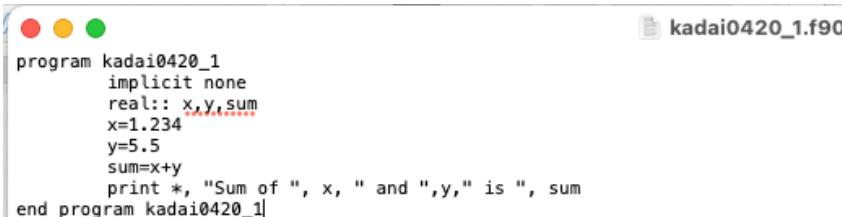
2.0 + 8 / 5 と 2 + 8.0 / 5

## 課題1

kadai0420.f90を実行する。

提出物：実行結果のスクリーンショット

ファイル名：kadai01\_1.png



```
● ● ●
program kadai0420_1
    implicit none
    real:: x,y,sum
    x=1.234
    y=5.5
    sum=x+y
    print *, "Sum of ", x, " and ",y," is ", sum
end program kadai0420_1
```

## 課題2

PRINT文を用いて、自分の学籍番号と名前（ローマ字）を表示するプログラムを作成、実行する。

提出物：プログラムファイル

ファイル名：kadai01\_2.f90

## 課題3

演算優先規則と整数型 (Integer) と実数型 (Real)  
を理解する。

提出物：実行結果のスクリーンショット

kadai01\_3.png

以下の計算式をPRINT文を用いて実行する

2.0 + 8 / 5

2 + 8.0 /5