2016/04/07更新

プログラミング演習

* 近年では、土木の分野においても、設計・製図やデータ処理など、あらゆる局面でコンピュータを用いることが多くなっており、コンピュータを必要に応じて道具として使いこなせる技術を身に着けてもらいたいと考えている。1年次の講義である「土木情報処理」では、表計算ソフト（Excel）やCADの基本操作に関する技量を身につけ、その次のステップとして本講義「プログラミング演習」では、プログラミング言語であるFortranを通じてプログラミングの基本的な技量を身につけてもらうことを目的とする。なお、Fortranは、社会的にはマイナーな言語と見られがちであるが、スーパーコンピュータを用いた科学技術計算では、現在もメジャーな言語である。
* Fortranを始めとしてプログラミング言語に関する教科書等書籍は数多くある。各自が生協等書店に赴き自らに適した書籍を購入されることを望んでいるため、本講義では教科書を指定しない。プログラミング技術を修得するためには、先ずは基本的な文法を簡単な問題を通じて修得することが重要であり、そのためにも自ら考え、手を動かすことが重要である。本講義は「演習」形式であるため、各自が積極的にチャレンジし、プログラミングの基礎的な技量を身につけ、卒業研究等に活かして欲しいと考えている。
* フォートランにはいくつかのバージョンがあり、本講義で使用するバージョンは、fortran77に準ずる。

# 2016年度　講義日（PC講義室１）

|  |  |  |  |
| --- | --- | --- | --- |
| No | 日付 | 内容 | 課題 |
| 1 | 4/11 | ガイダンスとセットアップ | レポート１ |
| 2 | 18 | プログラムの基本知識、行番号、コメント文、整数と実数 | レポート２ |
| 3 | 25 | 倍精度実数型、文字変数 | レポート３ |
| 4 | 5/9 | 変数名の付け方、式・代入、複素数 | レポート４ |
| 5 | 16 | 簡単な計算、組み込み関数 | レポート５ |
| 6 | 23 | 入出力（キーボード、ディスプレイ） | レポート６ |
| 7 | 30 | DOループ | レポート７ |
| 8 | 6/6 | 配列（その１） | レポート８ |
| 9 | 13 | 配列（その２） | レポート９ |
| 10 | 20 | IF文 | レポート１０ |
| 11 | 27 | ファイル操作 | レポート１１ |
| 12 | 7/4 | SOBROUTINE文 | レポート１２ |
| 13 | 11 | 最終レポートの説明 | 最終レポート |
| 14 | 18（祝） | レポート作成 |  |
| 15 | 25 | レポート提出日（18時までに提出すること） |  |

評価方法

7点×12レポート（レポート1～12）+16点×1レポート（最終レポート）=100点

※レポートは次回の講義終了後に提出

　なお、

**実習室での飲食は厳禁であり、ルールを守れない者は履修を認めない。**

引用・参考文献

・戸川隼人：演習と応用 FORTRAN77、サイエンス社、２００３．

・浦昭二編：FORTRAN77入門、培風館、２００８．

# 目次

[2015年度　講義日（PC講義室１） 1](#_Toc416188913)

[目次 2](#_Toc416188914)

[第1回：ガイダンスとセットアップ 5](#_Toc416188915)

[（Windowsの環境設定） 5](#_Toc416188916)

[（秀丸（テキストエディター）の設定） 5](#_Toc416188917)

[（Tera Term Pro(端末エミュレータでLinux上のコマンドが実行できる)の設定） 6](#_Toc416188918)

[（簡単なプログラムの作成） 9](#_Toc416188919)

[（コンパイル） 10](#_Toc416188920)

[演習 11](#_Toc416188921)

[レポート１ 11](#_Toc416188922)

[レポートの書き方 11](#_Toc416188923)

[第2回：プログラムの基本知識、行番号、コメント文、整数と実数 13](#_Toc416188924)

[前回からの変更点 13](#_Toc416188925)

[前回の復習 13](#_Toc416188926)

[2-1 プログラムの基本構造 13](#_Toc416188927)

[2-2 行（文）番号 13](#_Toc416188928)

[2-3 コメント文 13](#_Toc416188929)

[2-4 整数と実数 14](#_Toc416188930)

[レポート２ 16](#_Toc416188931)

[第３回　倍精度実数型、文字変数 17](#_Toc416188932)

[前回の補足 17](#_Toc416188933)

[前回の復習 17](#_Toc416188934)

[3-1　倍精度実数型の別の宣言方法 17](#_Toc416188935)

[演習 17](#_Toc416188936)

[3-2 文字変数 17](#_Toc416188937)

[演習 18](#_Toc416188938)

[レポート３ 18](#_Toc416188939)

[第４回　変数名の付け方、式・代入、複素数 19](#_Toc416188940)

[4-1 変数名の付け方 19](#_Toc416188941)

[4-2 式・代入 19](#_Toc416188942)

[4-3 複素数 19](#_Toc416188943)

[演習 20](#_Toc416188944)

[レポート４ 21](#_Toc416188945)

[第５回　簡単な計算、組み込み関数 22](#_Toc416188946)

[5-1 簡単な計算（復習） 22](#_Toc416188947)

[5-2 組み込み関数 22](#_Toc416188948)

[演習 25](#_Toc416188949)

[レポート５ 25](#_Toc416188950)

[第６回　入出力（キーボード、ディスプレイ） 26](#_Toc416188951)

[6-0 前回の補足説明 26](#_Toc416188952)

[6-1 入出力 26](#_Toc416188953)

[6-2 出力 26](#_Toc416188954)

[6-2-1 書式なし出力 26](#_Toc416188955)

[6-2-2 書式付き出力 27](#_Toc416188956)

[例題（F型、E型編集記述子） 28](#_Toc416188957)

[6-3 入力 29](#_Toc416188958)

[6-3-1 書式なし入力 29](#_Toc416188959)

[6-3-2 書式付き入力 29](#_Toc416188960)

[演習 29](#_Toc416188961)

[レポート6 29](#_Toc416188962)

[第７回 DOループ 31](#_Toc416188963)

[7-1 DO文によるループ 31](#_Toc416188964)

[演習 31](#_Toc416188965)

[7-2 DO文による二重ループ 32](#_Toc416188966)

[7-3 その他の形式 33](#_Toc416188967)

[7-3-1　行番号の省略 33](#_Toc416188968)

[7-3-2　整数だけでなく実数でもできる 34](#_Toc416188969)

[レポート7 35](#_Toc416188970)

[第8回　配列（1次元）（整数、実数） 36](#_Toc416188971)

[8-1 配列の宣言 36](#_Toc416188972)

[8-2 配列の計算 37](#_Toc416188973)

[8-3 文字列の配列 38](#_Toc416188974)

[レポート8 38](#_Toc416188975)

[第9回 配列（その２） 39](#_Toc416188976)

[9-1　2×2の行列式を求めるプログラム 39](#_Toc416188977)

[9-2 　2×2の逆行列を求めるプログラム 39](#_Toc416188978)

[レポート9 41](#_Toc416188979)

[第10回 IF文 42](#_Toc416188980)

[10-1 IF文 42](#_Toc416188981)

[10-2 GOTO文 43](#_Toc416188982)

[10-3 IFブロック 43](#_Toc416188983)

[レポート10 45](#_Toc416188984)

[第11回　ファイル操作 46](#_Toc416188985)

[レポート11 48](#_Toc416188986)

[第12回　SUBROUTINE 49](#_Toc416188987)

[レポート12 52](#_Toc416188988)

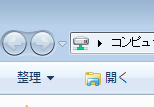
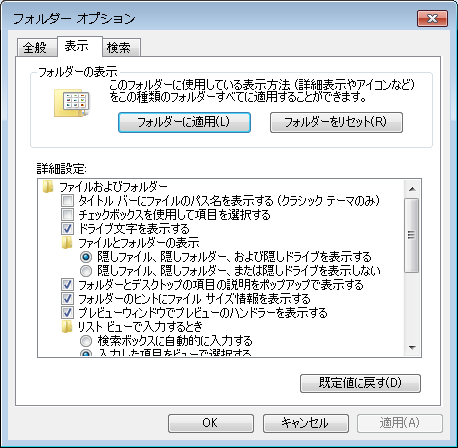
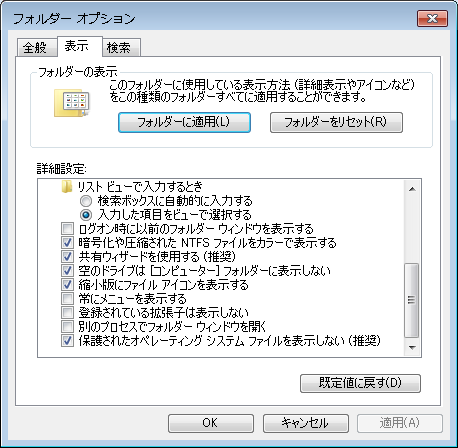
[第13回　最終レポート課題（レポート１３） 53](#_Toc416188989)

# 第1回：ガイダンスとセットアップ

* 秀丸をタスクバーに入れておくと便利。
* hello.fを選択し、マウスの右ボタンからプロパティを開き、ファイルの種類の変更で秀丸を指定すると、以降、拡張子ｆのファイルは、ファイルをダブルクリックすると開けるようになる。

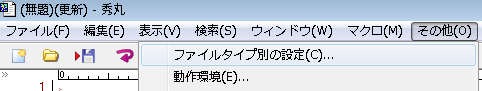
1. Windowsを起動する。

## （Windowsの環境設定）

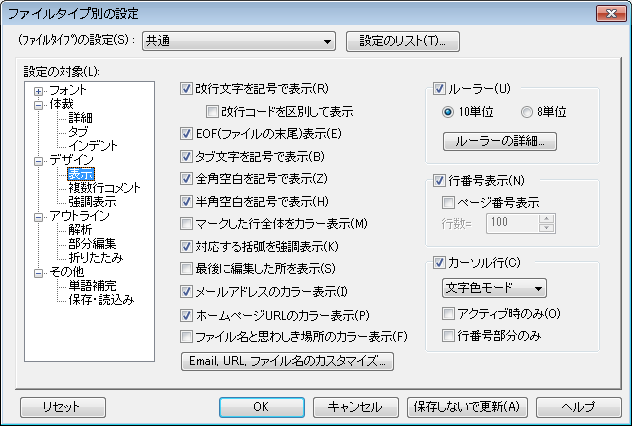
1. デスクトップのをダブルクリックし、エクスプローラを開く。
2. 右クリック→新規作成→フォルダーの操作で、新しいフォルダーを作成し、名前を”PROGRAM”(半角英文字)とする。
3. の整理から、“フォルダーと検索のオプション”を選択する。
4. を下にスクロールして、　□登録されている拡張しは表示しない、のチェックを外す。

## （秀丸（テキストエディター）の設定）

1. スタートボタンを押して、を探し、起動させる。
2. メニューから、その他を選択し、



一番上の“ファイルタイプ別の設定”を開く。

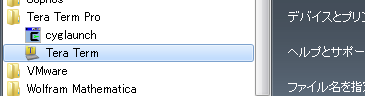


左側のデザイン、表示を選択し、左の赤枠のところにチェックを入れる。

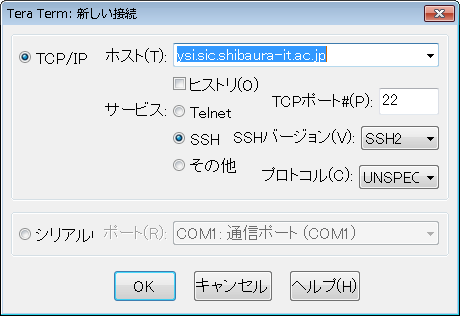
1. 秀丸のウィンドウで、半角スペース、全角スペース、タブを入力して、

表示されることを確認する。

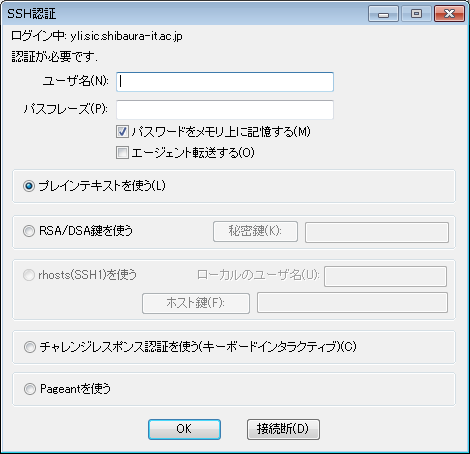
## （Tera Term Pro(端末エミュレータでLinux上のコマンドが実行できる)の設定）

1. スタートメニューから、

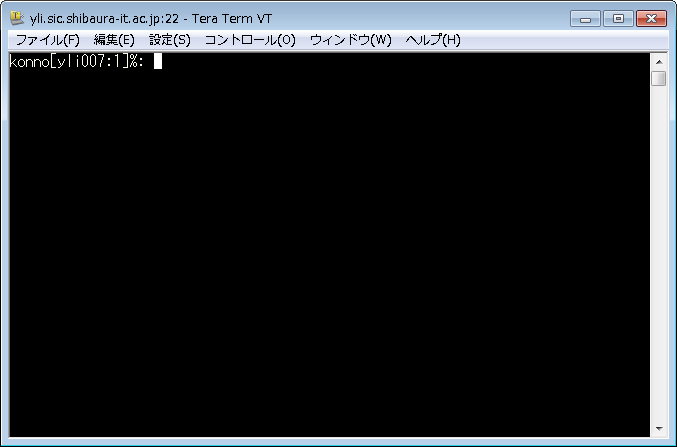
Tera Termを選択する。

1. 

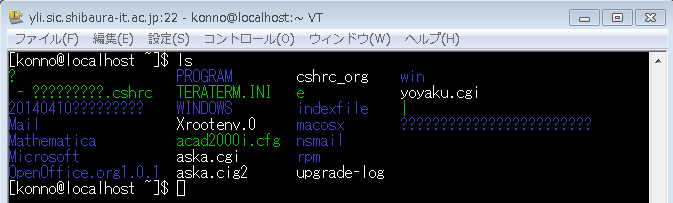
ホストに、ysi.sic.shibaura-it.ac.jpを指定し、OKボタンをおす。

1. 

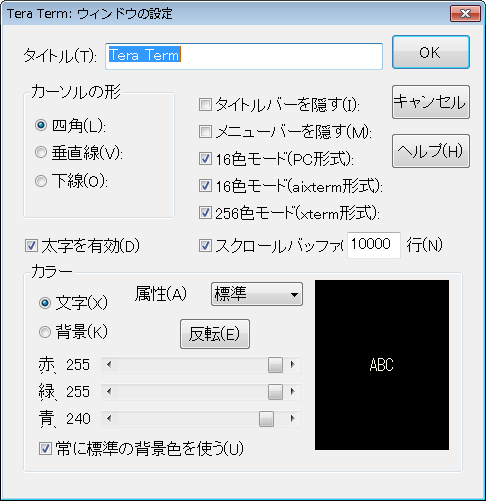
ユーザ名に学籍番号、パスフレーズに学情のパスワード(ガソット)を入力し、OKボタンを押す

1. 

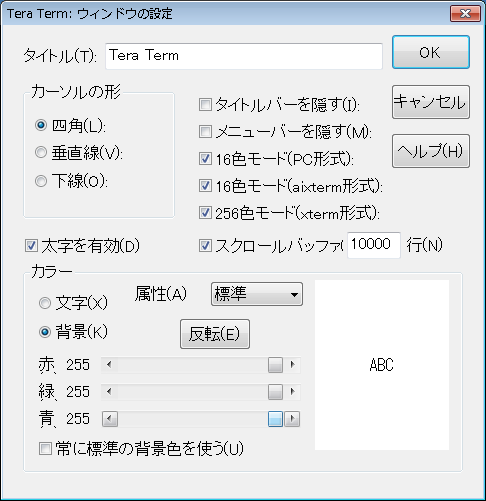
が開けばOK。

1. この黒い画面が、学情のLinuxマシンの端末になっている。以下のコマンドを記入して、エンターキーを押す。なお、Linuxでは大文字、小文字が区別されるので注意すること。
2. 

lsと入力し、エンターキーを押すと、今自分がいるディレクトリ（ホーム ディレクトリ）の中に存在しているファイルとディレクトリが表示される。（ディレクトリとはWindowsのフォルダに対応）

1. 先ほど、WindowsでPROGRAMというフォルダを作成したが、lsで現れたPROGRAMは同一である。（？？？は日本語フォントの部分で、このように文字化けしてしまう）
2. 次に、メニューから設定→ウィンドウを選択
3. 

※左の反転ボタンでもOK

1. 反転ボタンを押し、

文字が選択されているときに、赤緑青、すべてが０になっていることを確認

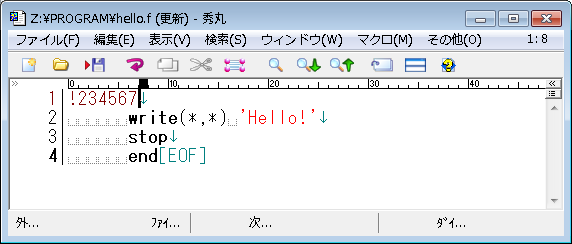
1. 背景を選択し、赤緑青をすべて255とする。
2. 一番下の□常に標準の背景色を使うのチェックを外すし、OKボタンを押す。
3. Tera Termのウィンドウの背景が白、文字が黒になったことを確認する。
4. 次に、メニューから設定→端末を選択し、

上図の赤枠のように設定する。

1. この設定を保存するため、メニューの設定→設定の保存で、設定を保存する(デスクトップなど、個人のファイルがおける場所に。例えば：設定の保存→デスクトップ→システムフォルダー（自分の学籍番号）→マイドキュメント)。
2. で開いたエクスプローラの中のファイルやフォルダと対応していることを確認すること。
3. 各自が作るプログラムは、PROGRAMの中に作ることにする。
4. Tera Termウィンドウは適当な大きさに変更し、画面の好みの場所に移動する。
5. なお、コマンドclearを使用すると、Tera Termウィンドウの表示がクリアされる。
6. コマンドpwdを使用すると、今いる場所（ディレクトリ）が確認できる。
7. ディレクトリの移動にはcdコマンドを使用する。
8. たとえば、ディレクトリPROGRAMへの移動は、”cd PROGRAM”でできる。
9. ここで、pwdを実行すると、PROGRAMに移動したことが確認できる。

.txtが表示されていないときには、上述の拡張子の設定が正しくない。

## （簡単なプログラムの作成）

1. では、簡単なプログラムを作ってみよう。
2. Windowsのエクスプローラで、フォルダPROGRAMを開き、右クリック→新規作成→テキスト作成を選択する。
3. 作成された“新しいテキスト ドキュメント.txt”の名前を”hello.f”とする。（※拡張子はforでも可）
4. 秀丸を起動する。エクスプローラでhello.fをマウスの左ボタンを使って、秀丸のウィンドウへドラッグ＆ドロップして、hello.fを開き、下記のように書き込む。
5. 
6. １行目の先頭の！は、同行の！以降の文字は、注釈であることを指示する。プログラムが長くなると、プログラムの流れなどがわかりにくくなるので、適当に注釈をいれるとよい。

|  |
| --- |
| 補足  ・列のことをフォートランではカラムと呼ぶ。  ・プログラムは７カラムから書く。  ・（よくあるミス）フォートランのプログラムには必ず、endが含まれている必要がある。したがって、endが書かれていない、エディターで書いていても保存していない状態（よくあるパターン。コンパイルする前に保存する習慣を！）では、コンパイル時にエラーがでるので注意。  ・！はc(小文字でも大文字でも可。なお、cはコメントの頭文字)  ・ただし、cは１カラム目でしか、使用できない。  ・一方、!はどのカラムでも使用でき、!以降は注釈文となる。  ・write,stop,endの行は、実行文と呼ぶ。  ・!は実行文の後ろにも書くことができるので、実行文の説明を書くのに便利に使える。  ・フォートランは大文字、小文字を区別しない |

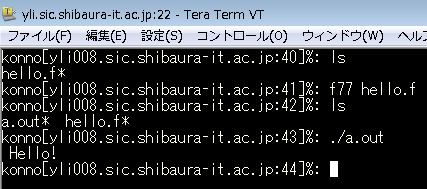
1. 書き込んだらファイルを保存する。（ファイルは、開いたままでも、閉じても、どちらでもかまわない）

補足：f90を実行して，以下の文字が表示され，うまくいかなかったら

f90: Command not found.

もう一度，tera termでysi.sic.shibaura-it.ac.jpに接続したかを確認すること．

## （コンパイル）

1. f90 hello.f
2. 上記の実行をコンパイル（プログラムソースを実行ファイルへ変換すること）という。hello.fの記述に文法上の間違いがあると、エラーメッセージがでる。その場合は、エラーメッセージなどを参考に、修正すること。
3. コンパイルが成功したら、コマンドlsを実行すること。実行ファイルa.outが生成されていることが分かる。
4. ./a.outと打ち込み、エンターキーを押すとHello!と表示される。
5. 

|  |
| --- |
| 補足  ・’/’はディレクトリ(場所)を意味している。./の.は自分を表している。  ・..は親ディレクトリを意味する。たとえば、ls ..を打ち込み、エンターキーを押すと、親のとろこにあるファイルとディレクトリが表示される。 |

## 演習

1. ‘Hello!’の中身を適当に変えて、同じように表示されるかを確認する。
2. ！のかわりにcを使ってみて、同じように実行されることを確認する。
3. write(\*,\*) 'Hello!' !あいさつ　　　のように、実行文の後ろに注釈をつけてみる。
4. write,stop,endのいずれかのスペルを間違えて、コンパイルしてみる。
5. cat hello.fを実行してみる。
6. !fを実行してみる。
7. f90 hello.f –o hello.out を実行してみる。Hello.outが生成されていることを確認し、./hello.outを実行してみる。（-oのoはアルファベット小文字のo）
8. f90 –o hello.out hello.f を実行してみる。Hello.outが生成されていることを確認し、./hello.outを実行してみる。(順番は問わず、-oの後ろのファイルが実行ファイルになる。また、実行ファイル名は基本的に何でもかまわないが、どのソースプログラムの実行ファイルか分からなくなるので、\*.fの\*を含んだ名前にすると良い。なお、hello.outは実行ファイルとなるが、名前は、hello.exeでも拡張子を外して、helloでも構わない。)

## レポート１

自分の名前、学籍番号を画面に表示させるプログラムを作成。このプログラムを印刷して提出（A4，1枚で）．

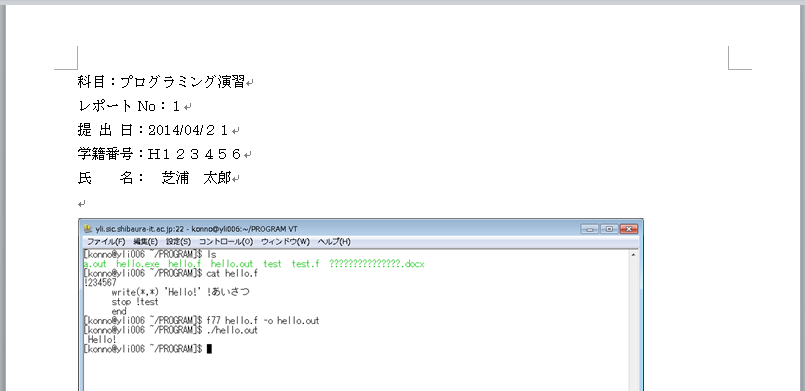
提出日：次回の講義終了後

※次回以降のレポートについては，その書き方については，次回説明する．

# 第2回：プログラムの基本知識、行番号、コメント文、整数と実数

## レポートの書き方

1. Tera Termの画面で、プログラムのあるフォルダまで行く。
   1. cd PROGRAM
   2. cd 01
   3. 補足説明 cd P\*でもOK。ワイルドカード(\*)が使える。毎回、プログラムを作成してもらうが、そのプログラムは、フォルダPROGRAMの下にフォルダ01,02,03,・・・を作成して、その中に入れる。（フォルダ01､02､・・・はエクスプローラの方で作成できる）
2. プログラムを作成し、正しく稼働したことを確認したら、以下の手順でレポートを作成すること。
3. まず、コマンドclearを実行
4. lsの実行
5. cat プログラム名（たとえば、cat hello.f）※catはファイルの中身をディスプレイ上に表示させるコマンド。
6. コンパイルの実行：f90 hello.f –o hello.out
7. ./hello.outの実行
8. Ctrl+Alt+PrtScnの３つのボタンを同時に押して、Tera Termウィンドウの表示内容をクリップボードにコピーする。（別方法：Tera tarmのウィンドウの中で、マウスの左ボタンで範囲を指定して、マウス右ボタンで選択した範囲のテキストがクリップボードに入る）
9. エクスプローラのウィンドウ上で右ボタンで新規作成でwordの文書を新規作成する。ファイル名は例えば、report01.docx、とする。
10. これをワードにコピーして、印刷したものを提出すること。（ワードの文書は残しておくこと）
11. なお、書式は、下図の通りとし、原則１枚のみとする。2ページにわたる場合には、両面印刷をして提出すること。（後ろの講義ではプログラムが長くなる可能性もあるので、その場合の対応については後日説明する）



## 前回からの変更点

□フォートランの実行文では、大文字小文字を区別しない。下記では、プログラムの本文は見やすい大文字で書くことにする。（なお，間違いやすい文字：l（Lの小文字）,1（数字の1）,I（iの大文字），0（ゼロ），o（英小文字オウ），O（英大文字オウ）など）

## 前回の復習

□Tera Termが起動できる

□ex01.fという名前で、今日は4月21日です、と表示するプログラムを作成し、コンパイルする。

## 2-1 プログラムの基本構造

FORTRANのプログラムでは、「(STOP)」「END」でプログラムの終了を宣言する。その間にプログラム本文を記載することが約束事。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | プログラム本文 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | STOP←省略可能。本文中のどこにでも書けるが、これに出くわすとプログラムはこの位置で強制終了する。 |
|  |  |  |  |  |  | END←省略不可能 |

## 2-2 行（文）番号

下記の例では、5行目の「WRITE(\*,100) AREA」でAREAを出力する形式（どのような桁数で記述するか）を行番号100のFORMAT文で指定している。なお、行番号は1～５列目に書く必要がある。

circle.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ! | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  |  |  |  |  | WRITE(\*,\*) 'INPUT RADIUS’ |
|  |  |  |  |  |  | READ(\*,\*) R |
|  |  |  |  |  |  | AREA=R\*R\*3.14 |
|  |  |  |  |  |  | WRITE(\*,100) AREA |
|  |  | **1** | **0** | **0** |  | FORMAT(F7.2) |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

行番号は必要に応じて付けるものであり、書式（FORMAT文）のほかに、分岐やジャンプなどで使います。行番号は必ずしも数字の小さいものから大きいものへと順番に付けていく必要はありません。

補足：FORMAT(F7.2)のF7.2は□□□□□□の升目に数字を入れることを指示している。ここで、F7.2の7は全体の升目の数、2は小数点以下の升目の数である。

## 2-3 コメント文

コメント文は、プログラムの中に記述できる注釈であり、多くの変数を使う場合にその変数が何を表わすのか、また、プログラムで何をしようとしているかを記述しておくと、後々見返す時に便利である。コメント文は[C]または[!]で始まる。FORTRANのバージョンにより異なるが、[C]は、行の一カラム目に記載する必要があり、［!］は行の途中にも置くことができる。

circle2.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | WRITE(\*,\*) 'INPUT RADIUS' |
| ! |  | Rは円の半径、AREAは円の面積である。 | | | | |
|  |  |  |  |  |  | READ(\*,\*) R !Rは変数 |
|  |  |  |  |  |  | AREA=R\*R\*3.14 !AREAも変数 |
|  |  |  |  |  |  | WRITE(\*,\*)AREA |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

また、二つ目の例のように、コメント文が2行以上ある場合には、例のように外見を整える場合もある。

コメント文が2行以上ある場合

circle3.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | WRITE(\*,\*) 'INPUT RADIUS' |
| c |  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | | |
| c |  | Rは円の半径 | | | | |
| c |  | AREAは円の面積 | | | | |
| c |  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | | |
|  |  |  |  |  |  | READ(\*,\*) R |
|  |  |  |  |  |  | AREA=R\*R\*3.14 |
|  |  |  |  |  |  | WRITE(\*,\*) AREA |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

## 2-4 整数と実数

　プログラムの中で値を保存しておくもの（値の記憶場所）を変数と呼びます。この変数に最初の値を代入することを初期化と呼びます。初期化していない（初期値を代入していない）変数の値は不定です。

　FORTRANで式の計算や変数の宣言を行うときには原則としてデータの型というものを意識する必要があります。以下に、本講義で使用するであろうデータ型の例を示しておく。それ以外にもあるので、参考書などで勉強すること。

|  |  |  |
| --- | --- | --- |
| 種類 | データ型 |  |
| 基本整数型 | INTEGER | -2,147,483,648～2,147,483,647（符号付き）の整数型 |
| 倍精度整数型 | INTEGER\*8 | -9,223,372,036,854,775,808～  9,223,372,036,854,775,807（符号付き）の整数型  9223372036854775807 |
| 単精度実数型 | REAL | 約6桁の精度を持つ浮動小数点型 |
| 倍精度実数型 | REAL\*8 | 約14桁の精度を持つ浮動小数点型 |
| 文字型 | CHARACTER |  |

（注意）

* 暗黙の型宣言（先頭の文字がI,J,K,L,M,Nのいずれかであれば整数型の変数の名前、それ以外は実数型の変数の名前であると暗黙のうちに決まってしまう）という約束がありますが、プログラム内で、データの型宣言をしていれば大丈夫である。この暗黙の型宣言はデータの型の宣言を省略してもプログラムを書くことができるようにするためにある。

|  |  |
| --- | --- |
| cat seisu-1.f  !234567  A=2.5  WRITE(\*,\*) A  N=2.5  write(\*,\*) N  STOP  END |  |

|  |  |
| --- | --- |
| cat seisu-2.f  !234567  INTEGER A,B  A=2.5  WRITE(\*,\*) A  STOP  END |  |

余裕があれば、seisu-3.fを打ち込んで、実行してみてください。

|  |  |
| --- | --- |
| cat seisu-3.f  !234567  N=2147483647  WRITE(\*,\*) N  N=N+1  WRITE(\*,\*) N  N=-2147483648  WRITE(\*,\*) N  N=N-1  WRITE(\*,\*) N  STOP  END | 隣同士 |

* 例えば単精度実数型の変数に[ 3.14159*265* ]を代入しても、[ 3.14159 ]までが記憶され、それ以後の[*265*]は変数に記憶されないことになる。従って高精度の計算が望まれる場合には倍精度実数型を使う。
* FORTRANの基本構造として、下記のように、プログラム本文にはまず型の宣言文があり、次に実行文が記載される。

circle4.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | REAL R, AREA　　　　　　　　　　　　　　　　（型宣言文） |
|  |  |  |  |  |  | WRITE(\*,\*) 'INPUT RADIUS'　　　　　　　　　　（以後、実行文） |
|  |  |  |  |  |  | READ(\*,\*) R |
|  |  |  |  |  |  | WRITE(\*,\*) R |
|  |  |  |  |  |  | AREA=R\*R\*3.14　 !単精度のときはこのまま、倍精度のときにはd0をつける |
|  |  |  |  |  |  | WRITE(\*,\*) ‘AREA=’, AREA |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

## レポート２

　上記プログラムをコンパイルし、実行し、Rに[ 1.23456789 ]を代入しせよ。

　また、同様に型を倍精度実数型(REAL\*8)に変更して実行し、Rに上記の値を代入してみよ。

レポートの内容

1. circle4.fを単精度実数型のプログラムとし,Rに1.23456789を代入する。この時のプログラムと面積の出力をレポートに書き込むこと。
2. circle4.fを倍精度実数型のプログラムとし（circle4.fをコピーして、名前をcircle5.fとする）,Rに1.23456789を代入する。この時のプログラムと面積の出力をレポートに書き込むこと。
3. レポートの表紙には、必ずプログラミング演習、レポート２、学籍番号、氏名、提出日を書くこと。

※2014年度はレポートの書き方の説明がこの時点でもまだたったので、上記のような提出方法となった。ただし、レポート０１で、既に、レポートの書き方に基づいて提出してくれており、書き方は上記でも、レポートの書き方に準じてもどちらでもOKとする。

# 第３回　倍精度実数型、文字変数

## 前回の補足

□[整数のところ](#_2-4__整数と実数)

## 前回の復習

□（2015年度ではこれまであまりいえていないが）変数には複数の種類がある。そのことに注意すること。

□フォルダPROGRAMの下に、フォルダ03を作成し、tera termでcdコマンドを使って03まで行く。

□ex02.fという名前で、三角形の面積を求めるプログラムを作成し、コンパイルする。

参考：

cat ex02.f

!234567

WRITE(\*,\*) '底辺の長さ=?'

READ(\*,\*) A

WRITE(\*,\*) '高さ=?'

READ(\*,\*) H

AREA=A\*H/2. !2は2.と小数点とする方がよい

WRITE(\*,\*) '底辺の長さ=',A

WRITE(\*,\*) '面積=',AREA

WRITE(\*,\*) '面積=',A\*H/2.

STOP

END

## 3-1　倍精度実数型の別の宣言方法

* 精度の高い数値計算を行う場合は、倍精度実数型を使うことを薦める。下記の宣言文を使うことにより、AからH、OからZで始まる変数は、すべて倍精度実数型となる。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | IMPLICIT REAL\*8 (A-H,O-Z) |

### 演習

|  |  |
| --- | --- |
| cat ex03-1.f  !234567  IMPLICIT REAL\*8(D)  !補足：上記は　REAL\*8 D1,D2　と書くこともできるが、IMPLICIT文を使うことにより、Dで始まる変数を一括で倍精度型にすることができる。  ! IMPLICIT REAL\*8(A-H,O-Z)  !補足：また、実数型変数を全て倍精度にするには、上記のように宣言すればよい。  D1=1.2345678901234567890  D2=1.2345678901234567890D0  WRITE(\*,100) D1  WRITE(\*,100) D2  100 FORMAT(F30.28)  STOP  END | 解説  実数の最後にD0を付けると精度が上がる。（D0は10の0乗という意味。例えば、D2であれば、10の２乗を表す）fはf型編集記述子と呼ばれる。例えば、f5.2は、□□.□□となり、□に１つの数字が入り、f5.2の5は小数点を含んだ全体の桁数で、２は小数点以下の桁数。なお、（プログラムは小文字で書いたけど、気にしないで） |
| cat ex03-2.f  !234567  WRITE(\*,\*) 1.d0  WRITE(\*,\*) 1.d2  WRITE(\*,\*) 1.d-2  STOP  END |  |

## 3-2 文字変数

* 文字型CHARACTERは、長さを指定して文字変数を宣言する。下記を実行してみなさい。

moji.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | CHARACTER C1\*5,C2\*8,C3\*14 |
|  |  |  |  |  |  | CHARACTER\*2 C1\_2,C2\_2 |
|  |  |  |  |  |  | C1='Civil' !5文字 |
|  |  |  |  |  |  | C2='ENGINEER' !8文字 |
|  |  |  |  |  |  | C3=C1//'+'//C2 !//で文字を結合することができる。 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C1=',C1 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C2=',C2 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C3=',C3 |
|  |  |  |  |  |  | WRITE(\*,\*) C1(2:4) |
|  |  |  |  |  |  | C1\_2=C1 |
|  |  |  |  |  |  | C2\_2=C2 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C1\_2=',C1\_2 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C2\_2=',C2\_2 |
|  |  |  |  |  |  | C1='C234567890' |
|  |  |  |  |  |  | WRITE(\*,\*) C1 !C1は5文字で宣言しているので、これを超える分は無視される |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

### 演習

|  |  |
| --- | --- |
| cat ex03-3.f  CHARACTER STR\*10  WRITE(\*,\*) '半角文字を１０文字入力してください'  READ(\*,\*) STR  WRITE(\*,\*) STR  WRITE(\*,'(A10)') STR  WRITE(\*,'(A5)') STR  WRITE(\*,'(A)') STR  STOP  END | 解説：WRITE(\*,\*) はフリーフォーマット。通常、実行すると２カラム目から表示される。Aの後ろは文字数を指定できる。文字数を省略した場合、宣言文の文字数となる。なお、AはA型編集記述子と呼ばれる。  補足：  入力する文字は何でもよいが、まず、  1234567890  を代入してみるとよい。 |

## レポート３

report03.fという名前で、下記の内容のプログラム（report03.f）を作成せよ。

・姓（name1）と名(name2)をキーボードで入力できるようにし、姓と名を結合したもの(name)を画面に出力する。

ヒント：name=name1//' '//name2

※名前は半角アルファベットを使用すること。

# 第４回　変数名の付け方、式・代入、複素数

連休明け初回なので、思い出しながら、ゆっくりいきましょう！

## 4-1 変数名の付け方

変数名には以下のルールがある．

・アルファベットの大文字，小文字は区別されない．

・使用できる記号は，アルファベット，数字，\_（アンダーバー），である．

・変数名の最初は，アルファベットである必要があるが、途中で数字や\_（アンダーバー）を使うことができる．

・ルールではないが，変数名を工夫することによってプログラムは非常に読みやすくなる．

例えば，t（時間）,x（ｘ座標値）,sum（合計）など

## 4-2 式・代入

FORTRANでは和差積商「＋－＊／（全て半角表記）」が使えます。なお、計算の優先順位は、通常の四則演算のルールと同じで、（\*/）が先で、(+-)が次である。

また、等号「＝」は「イコール」という意味ではなく、「右辺の値を左辺に代入する」という意味です。

shisoku.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | L=3 ! Lに3を代入 |
|  |  |  |  |  |  | M=4 !Mに4を代入 |
|  |  |  |  |  |  | N=L\*M !NにL(=3)とM(=4)の積の値(12)を代入 |
|  |  |  |  |  |  | N=N+1 !N(=12)と1の和の値(=13)をNに再度代入 |
|  |  |  |  |  |  | WRITE(\*,\*) 'N=',N |
|  |  |  |  |  |  | A=L/M !右辺はすべて整数なので、左辺には整数の値が代入される。なお、小数点以下は切り捨て。 |
|  |  |  |  |  |  | WRITE(\*,\*) 'A=',A |
|  |  |  |  |  |  | B=1.\*L/M |
|  |  |  |  |  |  | WRITE(\*,\*) 'B=',B |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

## 4-3 複素数

|  |  |  |
| --- | --- | --- |
| 種類 | データ型 |  |
| 単精度複素数型 | COMPLEX | 約6桁の精度を持つ浮動小数点型 |
| 倍精度複素数型 | COMPLEX\*16 | 約14桁の精度を持つ浮動小数点型 |

fukuso-1.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | IMPLICIT COMPLEX(C) |
|  |  |  |  |  |  | COMPLEX i |
|  |  |  |  |  |  | i=(0.,1.) !（実部,虚部）でどちらも数字を入れる必要がある。変数名で入れることはできない。 |
|  |  |  |  |  |  | WRITE(\*,\*) 'i=',i |
|  |  |  |  |  |  | C1=2.\*i |
|  |  |  |  |  |  | C2=i\*i |
|  |  |  |  |  |  | WRITE(\*,\*) 'C1=',C1 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C2=',C2 |
|  |  |  |  |  |  | WRITE(\*,\*) 'C1+C2=', C1+C2 |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

### 演習

|  |  |
| --- | --- |
| cat fukuso-2.f  COMPLEX i  COMPLEX\*16 i2  i=(0.0,1.0)  i2=(0.0d0,1.0d0)  WRITE(\*,\*) i  WRITE(\*,\*) i2  WRITE(\*,100) i  WRITE(\*,100) i2  100 format(2f20.17)  WRITE(\*,100) i\*i  WRITE(\*,100) i2\*i2  STOP  END | 解説：2f20.17の最初の２は、２つの実数をフォーマットとなっており、複素数は２つの実数で構成されていることに注意。 |
| cat fukuso-3.f  COMPLEX Z  COMPLEX\*16 Z2  Z=(3.,4.)  Z2=(3.D0,4.D0)  WRITE(\*,\*) CABS(Z) !複素数の大きさ（引数が単精度の場合）  WRITE(\*,\*) CDABS(Z2) !複素数の大きさ（引数が倍精度の場合）  WRITE(\*,\*) REAL(Z) !実部の取り出し（引数が単精度の場合）  WRITE(\*,\*) DBLE(Z2) !実部の取り出し（引数が倍精度の場合）  WRITE(\*,\*) AIMAG(Z) !虚部の取り出し（引数が単精度の場合）  WRITE(\*,\*) DIMAG(Z2) !虚部の取り出し（引数が倍精度の場合）  WRITE(\*,\*) CONJG(Z) !共役複素数（引数が単精度の場合）  WRITE(\*,\*) DCONJG(Z2)!共役複素数（引数が倍精度の場合）  STOP  END |  |
| cat fukuso-4.f  COMPLEX Z  COMPLEX\*16 Z2  REAL\*8 X2,Y2  X=3.  Y=4.  Z=CMPLX(X,Y) !引数に変数が使える.  WRITE(\*,\*) Z  WRITE(\*,\*) CMPLX(X,0.) !  X2=3.D0  Y2=4.D0  Z2=DCMPLX(X2,Y2) !引数に変数が使える.  WRITE(\*,\*) Z2  WRITE(\*,\*) DCMPLX(X2,0.D0) !  STOP  END |  |

## レポート４

単精度複素数変数Z1,Z2を用意し、Z1=3+4iとZ2=3-4iとする。次の計算を行うプログラム(report04.f)を作成せよ。

・足し算（Z1+Z2）

・引き算（Z1-Z2）

・掛け算（Z1\*Z2）

・割 算（Z1/Z2）

# 第５回　簡単な計算、組み込み関数

## 5-1 簡単な計算（復習）

例題(ex05-1.f)

!234567

A=1.5

B=0.5

C=2.5

D=3.5

WRITE(\*,\*) 'A=',A

WRITE(\*,\*) 'B=',B

WRITE(\*,\*) 'A+B=',A+B

WRITE(\*,\*) 'A-B=',A-B

WRITE(\*,\*) 'A/B=',A/B

WRITE(\*,\*) 'A\*B=',A\*B

WRITE(\*,\*) 'A+B\*C-D=',A+B\*C-D

WRITE(\*,\*) 'A+(B\*C)-D=',A+(B\*C)-D !演算の順番を変えるには、

WRITE(\*,\*) '((A+B)\*C)-D=',((A+B)\*C)-D !かっこ()が使用できる

STOP

END

例題（ex05-2.f）

!234567

REAL\*8 D

WRITE(\*,\*) '2.\*2.\*2.=',2.\*2.\*2. !定数は小数点を付けて書くのが良い

WRITE(\*,\*) '2.1\*2.1\*2.1=',2.1\*2.1\*2.1

WRITE(\*,\*) '2.1\*\*3.=',2.1\*\*3. !\*\*はべき乗演算

WRITE(\*,\*) '2.1D0\*\*3.=',2.1D0\*\*3. !D0を付けると倍精度の計算になる

D=2.1

WRITE(\*,\*) 'D=',D

WRITE(\*,\*) 'D\*\*3.=',D\*\*3. !Dは倍精度であるが、倍精度で入力されていない

D=2.1D0 !D0を付けることにより、変数Dに倍精度の値が設定される

WRITE(\*,\*) 'D=',D

WRITE(\*,\*) 'D\*\*3.=',D\*\*3. !倍精度で計算されたことが確認される

STOP

END

## 5-2 組み込み関数

FORTRANには多くの関数が組み込まれている。その一部を以下に紹介します。一部ですので他に多くの組み込み関数があります。参考書などで必要に応じて調べてください。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | 組込み関数 | 総称名 | 個別名 | 引数個数 | 引数型 | 関数型 | 使用例の説明 |
| 1 | 絶対値 | ABS | IABS  ABS  DABS | 1 | 整  実  倍 | 引数と同じ | ABS（I） |
| 2 | 切り捨て | INT | AINT  DINT | 1 | 実  倍 | 引数と同じ | AINT（R） |
| 3 | 剰余 | MOD | MOD  AMOD  DMOD | 2 | 整  実  倍 | 引数と同じ | MOD(3,2)  3を2で割った余り1を返す |
| 4 | 平方根 | SQRT | SQRT  DSQRT | 1 | 実  倍 | 引数と同じ | SQRT（R） |
| 5 | 指数 | EXP | EXP  DEXP | 1 | 実  倍 | 引数と同じ | EXP（R） |
| 6 | 自然対数 | LOG | ALOG  DLOG | 1 | 実  倍 | 引数と同じ | LOG（R） |
| 7 | 常用対数 | LOG10 | ALOG10  DLOG10 | 1 | 実  倍 | 引数と同じ | ALOG10（R） |
| 8 | 正弦 | SIN | SIN  DSIN | 1 | 実  倍 | 引数と同じ | SIN（R） |
| 9 | 余弦 | COS | COS  DCOS | 1 | 実  倍 | 引数と同じ | COS（R） |
| 10 | 正接 | TAN | TAN  DTAN | 1 | 実  倍 | 引数と同じ | TAN（R） |
| 11 | 逆正弦 | ASIN | ASIN  DASIN | 1 | 実  倍 | 引数と同じ | ASIN（R） |
| 12 | 逆余弦 | ACOS | ACOS  DACOS | 1 | 実  倍 | 引数と同じ | ACOS（R） |
| 13 | 逆正接 | ATAN | ATAN  DATAN | 1 | 実  倍 | 引数と同じ | ATAN（R） |
| 14 | 最大値 | MAX | AMAX0  AMAX1  DMAX1 | 2つ以上 | 整  実  倍 | 実  実  倍 | AMAX0(n1,n2[,n3,･･･])  AMAX1(a1,a2[,a3,･･･])  DMAX1(d1,d2[,d3,･･･]) |
| 15 | 最小値 | MIN | AMIN0  AMIN1  DMIN1 | 2つ以上 | 整  実  倍 | 実  実  倍 | AMIN0(n1,n2[,n3,･･･])  AMIN1(a1,a2[,a3,･･･])  DMIN1(d1,d2[,d3,･･･]) |

例題(ex05-3.f)

!234567

REAL A

REAL\*8 D

!01-----------------------------------------------------

N=-2

A=-2.

D=-2.D0

WRITE(\*,\*) 'N=',N,' ---> ','IABS(N)=',IABS(N)

WRITE(\*,\*) 'A=',A,' ---> ','ABS(A) =',ABS(A)

WRITE(\*,\*) 'D=',D,' ---> ','DABS(D)=',DABS(D)

WRITE(\*,\*)

!02-----------------------------------------------------

A=2.999

D=2.999D0

WRITE(\*,\*) 'A=',A,' ---> ','AINT(A)=',AINT(A)

WRITE(\*,\*) 'D=',D,' ---> ','DINT(D)=',DINT(D)

WRITE(\*,\*)

!03-----------------------------------------------------

N=3

A=3.5

D=3.5D0

WRITE(\*,\*) 'N=',N,' ---> ','MOD(N,2)=',MOD(N,2)

WRITE(\*,\*) 'A=',A,' ---> ','AMOD(A,1.7)=',AMOD(A,1.7)

WRITE(\*,\*) 'D=',D,' ---> ','DMOD(D,1.7D0)=',DMOD(D,1.7D0)

WRITE(\*,\*)

!04-----------------------------------------------------

A=4.0

D=4.0D0

WRITE(\*,\*) 'A=',A,' ---> ','SQRT(A)=',SQRT(A)

WRITE(\*,\*) 'D=',D,' ---> ','DSQRT(D)=',DSQRT(D)

WRITE(\*,\*)

!05-----------------------------------------------------

A=1.0

D=1.0D0

WRITE(\*,\*) 'A=',A,' ---> ','EXP(A)=',EXP(A)

WRITE(\*,\*) 'D=',D,' ---> ','DEXP(D)=',DEXP(D)

WRITE(\*,\*)

!06-----------------------------------------------------

A=10.0

D=10.0D0

WRITE(\*,\*) 'A=',A,' ---> ','ALOG(A)=',ALOG(A)

WRITE(\*,\*) 'D=',D,' ---> ','DLOG(D)=',DLOG(D)

WRITE(\*,\*)

!07-----------------------------------------------------

A=10.0

D=10.0D0

WRITE(\*,\*) 'A=',A,' ---> ','ALOG10(A)=',ALOG10(A)

WRITE(\*,\*) 'D=',D,' ---> ','DLOG10(D)=',DLOG10(D)

WRITE(\*,\*)

!08-----------------------------------------------------

A=3.14

D=3.14D0

WRITE(\*,\*) 'A=',A,' ---> ','COS(A)=',COS(A)

WRITE(\*,\*) 'D=',D,' ---> ','DCOS(D)=',DCOS(D)

WRITE(\*,\*)

!09-----------------------------------------------------

A=3.14

D=3.14D0

WRITE(\*,\*) 'A=',A,' ---> ','SIN(A)=',SIN(A)

WRITE(\*,\*) 'D=',D,' ---> ','DSIN(D)=',DSIN(D)

WRITE(\*,\*)

!10-----------------------------------------------------

A=3.14

D=3.14D0

WRITE(\*,\*) 'A=',A,' ---> ','TAN(A)=',TAN(A)

WRITE(\*,\*) 'D=',D,' ---> ','DTAN(D)=',DTAN(D)

WRITE(\*,\*)

!11-----------------------------------------------------

A=1.0

D=1.0D0

WRITE(\*,\*) 'A=',A,' ---> ','ACOS(A)=',ACOS(A)

WRITE(\*,\*) 'D=',D,' ---> ','DACOS(D)=',DACOS(D)

WRITE(\*,\*)

!12-----------------------------------------------------

A=1.0

D=1.0D0

WRITE(\*,\*) 'A=',A,' ---> ','ASIN(A)=',ASIN(A)

WRITE(\*,\*) 'D=',D,' ---> ','DASIN(D)=',DASIN(D)

WRITE(\*,\*)

!13-----------------------------------------------------

A=1.0

D=1.0D0

WRITE(\*,\*) 'A=',A,' ---> ','ATAN(A)=',ATAN(A)

WRITE(\*,\*) 'D=',D,' ---> ','DATAN(D)=',DATAN(D)

WRITE(\*,\*)

STOP

END

例題(ex05-4.f)

!234567

INTEGER N1,N2,N3 !省略可能

REAL A1,A2,A3 !省略可能

REAL\*8 D1,D2,D3

!14-----------------------------------------------------

N1=1

N2=2

N3=3

A1=1.;A2=2.;A3=3. !セミコロン(;)を使うと、実行文を続けて書くことができる

D1=1.;D2=2.;D3=3.

WRITE(\*,\*) 'N1=',N1,' N2=',N2,' N3=',N3,' ---> AMAX0(N1,N2,N3)=' !実行文は72カラム以内に書く必要がある。

$,AMAX0(N1,N2,N3) !これを超える場合には、6カラムに一文字

WRITE(\*,\*) 'A1=',A1,' A2=',A2,' A3=',A3,' ---> AMAX1(A1,A2,A3)=' !（0(ゼロ)、空白以外）を書き込めば、

$,AMAX1(A1,A2,A3) !継続行とみなされる。

WRITE(\*,\*) 'D1=',D1,' D2=',D2,' D3=',D3,' ---> DMAX1(D1,D2,D3)='

$,DMAX1(D1,D2,D3)

WRITE(\*,\*)

!15-----------------------------------------------------

WRITE(\*,\*) 'N1=',N1,' N2=',N2,' N3=',N3,' ---> AMIN0(N1,N2,N3)='

$,AMIN0(N1,N2,N3)

WRITE(\*,\*) 'A1=',A1,' A2=',A2,' A3=',A3,' ---> AMIN1(A1,A2,A3)='

$,AMIN1(A1,A2,A3)

WRITE(\*,\*) 'D1=',D1,' D2=',D2,' D3=',D3,' ---> DMIN1(D1,D2,D3)='

$,DMIN1(D1,D2,D3)

STOP

END

## 演習

1. 例題(ex05-3.f, ex05-4.f)の引数の値を変えて、計算結果が予想通りになるかを確認せよ。
2. 組込み関数は指定された引数型を使用しないとコンパイル時にエラーがでたり、計算が正しく行われないことがある。いくつかの関数でこのことを確かめよ。（たとえば、Aは単精度の変数として、dcos(A)とした場合、どうなるか？）
3. 三角関数の定義よりcos(π)=-1である。したがって、π=cos-1(-1)である。この式と組み込み関数を使って、πを求めるプログラム(pi.f)を作成せよ.(単精度と倍精度の両方で実行して、比べてみるとよい)

## レポート５

tan(pi/4)=1である。この式と組み込み関数を使って、πを求めるプログラム(report05.f)を作成せよ. (単精度と倍精度の両方で作ること)

# 第６回　入出力（キーボード、ディスプレイ）

## 6-0 前回の補足説明

a-h,o-zで始まる変数は、自動的に単精度実数型となる。したがって、単精度実数型として使う場合は、宣言する必要はないが、明確に示しておきたい場合には、宣言文で書いてもよい。

例　変数a1,a2,a3を単精度実数型として扱う場合の例。

ex06-0.f

!234567

REAL\*4 A1,A2,A3 !\*4は省略できる。また、A1,A2,A3のいずれも、自動的に単精度実数型なので、

A1=1. !REAL\*4自体も省略できる。

A2=2.

A3=A1+A2

WRITE(\*,\*) A3

STOP

END

## 6-1 入出力

ここでは、基本的な出力コマンド [ WRITE ]と入力コマンド [ READ ]の使用について整理する。

基本的なコマンド使用方法は

　　　　　READ(mm,nn)

　　　　　WRITE(mm,nn)

である。各コマンドの後に続く引数(mm,nn)は意味があります。

|  |  |
| --- | --- |
| mm | 入出力を行う場合に利用する装置に割り当てられた番号を明確に指定するのが本来の書き方である。近年のコンピュータでは  　　　　5------キーボード　　　　　　6------ディスプレイ  であり、キーボード入力し、ディスプレイに出力するには  　　　　　READ(\*,\*)  　　　　　WRITE(\*,\*)  となる。  但し、キーボード入力、ディスプレイ出力は標準入出力装置であり、装置番号のかわりにアスタリスク(\*)を用いることが一般的です。  それ以外にファイルからの入力やファイルへの出力などを行う場合にはファイル番号を記述する（後述） |
| nn | 入出力の書式を指定した行番号を記載するものであり、書式なしであればアスタリスク＊を用いる。 |

## 6-2 出力

### 6-2-1 書式なし出力

　ディスプレイに書式なしに出力する場合には「WRITE(\*,\*)」を用いることになる。

ex06-1.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGER L,M |
|  |  |  |  |  |  | L=3 |
|  |  |  |  |  |  | M=4 |
|  |  |  |  |  |  | WRITE(\*,\*)’ Result of Calculation’ |
|  |  |  |  |  |  | WRITE(\*,\*)L, M, L+M |
|  |  |  |  |  |  | WRITE(\*,\*)’L=’, L,’ M=’,M,’ Then L+M=’,L+M |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

### 6-2-2 書式付き出力

　前節の書式なし出力は、出力書式をコンパイラに任せて、とにかく値を何らかの意味のある書式で出力したい時に便利ですが、書式を指定する必要がある場合もありますので、書式付き出力の方法について学んでおきましょう。

　書式出力は一般的にFORMAT文を用います。

　WRITE文の（　）内の2番目の引数にFORMAT文の行番号を記入します。

ex06-2.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | REAL\*8 R, AREA |
|  |  |  |  |  |  | WRITE(\*,\*) ’INPUT RADIUS’ |
|  |  |  |  |  |  | READ(\*,\*) R |
|  |  |  |  |  |  | AREA=R\*R\*3.14D0 |
|  |  |  |  |  |  | WRITE(\*,**100**)　R,AREA　　　　　　　　　　行番号100の書式に従い出力 |
|  |  | **1** | **0** | **0** |  | FORMAT(F10.2, F10.2) |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

　ここで示されるF7.2は

最大7桁、小数点以下2桁の書式で実数を出力する

　ことを意味している。

　このFのような編集記述子の例を以下に示しておく。編集記述子については他にもあるので各自参考書などで勉強しておくこと。

|  |  |  |
| --- | --- | --- |
| 記号 | データの型 |  |
| I | 整数型 | 整数を小数点の無い整数形式で出力する。  「In」という書式で、最大n桁の整数を整数を示す |
| F | 実数型 | 実数を小数点のある実数形式で出力する。「Fn.m」という書式で、符号(+/-)を含む出力の最大幅n桁、小数点以下m桁 |
| E | 実数型 | 実数をnnE+mmの形式で出力する。「En.m」という書式で符号(+/-)やEを含む最大の出力幅n桁、精度がm桁ということになる。 |
| X | 空白 | 「nX」という書式で空白n個を出力する |
| A | 文字型 | 「An」といいう形式でnで文字数を指定する。ｎを省略した場合、宣言文で指定された文字変数の文字数が適用される。  （[ex03-3.fを参照](#_演習)） |

　なお[I3]という書式で、「12345」という整数値を出力しようとすると、「\*\*\*\*\*」というアスタリスクが表示されることになる。すなわち指定した書式に数値がおさまらない場合にはアスタリスクの列が表示される。

ex06-3.f

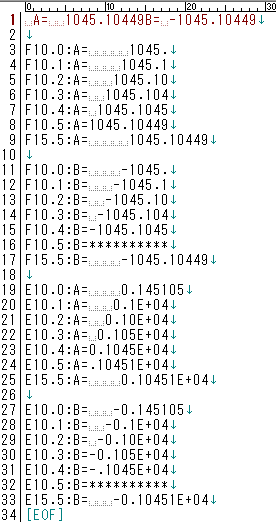
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGER M,N |
|  |  |  |  |  |  | M=123 |
|  |  |  |  |  |  | N=12345 |
|  |  |  |  |  |  | WRITE(\*,100)M,N |
|  |  |  |  |  |  | WRITE(\*,200)M,N |
|  |  |  |  |  |  | WRITE(\*,300)M,N |
|  |  |  |  |  |  | WRITE(\*,400)M,N |
|  |  | **1** | **0** | **0** |  | FORMAT(I5,I5) |
|  |  | **2** | **0** | **0** |  | FORMAT(2I5) |
|  |  | **3** | **0** | **0** |  | FORMAT(I5,1x,I5) |
|  |  | **4** | **0** | **0** |  | FORMAT(2(1x,I5)) |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

なお、出力FORMATは、WRITE文の中にもFORMAT文を用いなくても記載することができる.

下記を実行して上記プログラムの例と比較しなさい。

ex06-4.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGER M,N |
|  |  |  |  |  |  | M=123 |
|  |  |  |  |  |  | N=12345 |
|  |  |  |  |  |  | WRITE(\*,’(I5,I5)’) M,N |
|  |  |  |  |  |  | WRITE(\*,’(2I5)’) M,N |
|  |  |  |  |  |  | WRITE(\*,’ (I5,X,I5)’) M,N !FORMAT文中では、1Xとしないとダメ。 |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |



### 例題（F型、E型編集記述子）

A=1045.1045

B=-A

OPEN(10,FILE='F\_E.out')

WRITE(10,\*)'A=',A,'B=',B

WRITE(10,'(/A,F10.0)')'F10.0:A=',A

WRITE(10,'(A,F10.1)') 'F10.1:A=',A

WRITE(10,'(A,F10.2)') 'F10.2:A=',A

WRITE(10,'(A,F10.3)') 'F10.3:A=',A

WRITE(10,'(A,F10.4)') 'F10.4:A=',A

WRITE(10,'(A,F10.5)') 'F10.5:A=',A

WRITE(10,'(A,F15.5)') 'F15.5:A=',A

WRITE(10,'(/A,F10.0)')'F10.0:B=',B

WRITE(10,'(A,F10.1)') 'F10.1:B=',B

WRITE(10,'(A,F10.2)') 'F10.2:B=',B

WRITE(10,'(A,F10.3)') 'F10.3:B=',B

WRITE(10,'(A,F10.4)') 'F10.4:B=',B

WRITE(10,'(A,F10.5)') 'F10.5:B=',B

WRITE(10,'(A,F15.5)') 'F15.5:B=',B

WRITE(10,'(/A,E10.0)')'E10.0:A=',A

WRITE(10,'(A,E10.1)') 'E10.1:A=',A

WRITE(10,'(A,E10.2)') 'E10.2:A=',A

WRITE(10,'(A,E10.3)') 'E10.3:A=',A

WRITE(10,'(A,E10.4)') 'E10.4:A=',A

WRITE(10,'(A,E10.5)') 'E10.5:A=',A

WRITE(10,'(A,E15.5)') 'E15.5:A=',A

WRITE(10,'(/A,E10.0)')'E10.0:B=',B

WRITE(10,'(A,E10.1)') 'E10.1:B=',B

WRITE(10,'(A,E10.2)') 'E10.2:B=',B

WRITE(10,'(A,E10.3)') 'E10.3:B=',B

WRITE(10,'(A,E10.4)') 'E10.4:B=',B

WRITE(10,'(A,E10.5)') 'E10.5:B=',B

WRITE(10,'(A,E15.5)') 'E15.5:B=',B

CLOSE(10)

STOP

END

## 6-3 入力

### 6-3-1 書式なし入力

　書式なしでキーボード入力する場合には「READ(\*,\*)」を用いることになる。

ex06-5.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGER M,N !省略可能 |
|  |  |  |  |  |  | READ(\*,\*)M,N |
|  |  |  |  |  |  | WRITE(\*,\*)M+N |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

ここでMに[3]、Nに[4]を代入したければ、「3 4」と入力する。

### 6-3-2 書式付き入力

　書式付きでキーボード入力する場合には、書式付き出力とほぼ同様の扱いをする。

ex06-6.f

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGER M,N !省略可能 |
|  |  |  |  |  |  | READ(\*,200)M |
|  |  |  |  |  |  | N=10 |
|  |  | **2** | **0** | **0** |  | FORMAT(I3) |
|  |  |  |  |  |  | WRITE(\*,\*)M+N |
|  |  |  |  |  |  | STOP |
|  |  |  |  |  |  | END |

上述の例は、整数変数Mに整数3桁で入力することを意味している。

## 演習

1. ex06-6.fを実行し、Mに12,123,1234をそれぞれ代入したときの違いを確かめなさい。
2. 次のex06-7.fを実行し、書式と出力の関係を確認せよ。

ex06-7.f

!234567

WRITE(\*,\*) '実数を２つ入力してください'

READ(\*,\*) A,B

WRITE(\*,'(F5.2,F5.2)') A,B

WRITE(\*,'(F5.2,10X,F5.2)') A,B

WRITE(\*,'(2F5.2)') A,B

WRITE(\*,'(F5.2/f5.2)') A,B !/により強制改行される。'(/F5.2////f5.2/)'の用に、先頭、末尾、複数も可能である。

STOP

END

## レポート6

台形の面積を求めるプログラム(report06.f)を作成せよ。仕様は以下の通りとする。

1. 実数は倍精度実数型とする。
2. 上辺の長さの変数をA,下辺の長さをB,高さをHとし、面積をAREAとする。
3. A,B,Hはキーボードで入力する。
4. 出力はディスプレイとする。
5. Aに１，Bに２，Hに３を代入したときにディスプレイに以下のように表示されるプログラムとせよ。

(ヒント：WRITE(\*,’(A)’) ‘ABC=’　あるいはWRITE(\*,’(A4)’) ‘ABC=’)

A=001.00

B=002.00

H=003.00

AREA=004.50

※小数点の前の０は空白（ブランク）とする。

# 第７回 DOループ

プログラムの中で、一連のコードを繰り返し実行したい時があります。

## 7-1 DO文によるループ

　あらかじめ分かっている回数だけ特定の作業を繰り返したい時には、DO文を使います。

　　　① 　DO　(DOブロックの最終行番号) I=S,E

行番号　実行文

　　　DO文はDO文のすぐ下の行から指定された最終行番号までを繰り返す。その際、カウンター変数Iの初期値S, 終了値Eを指定し、Iの値を1ずつ増加させながら終了値Eになるまで繰返しを行う。

ex07-1.f

INTEGER TOTAL

TOTAL=0 !TOTALの初期値として0

DO 10 I=1,10 !Iを1から10まで1ずつ増やしながら行番号10までを実行する命令

10 TOTAL=TOTAL+I !TOTALにIを足しTOALに代入

WRITE(\*,\*) 'TOTAL=',TOTAL

STOP

END

　　　② 　DO　(DOブロックの最終行番号) I=S,E

　　　　実行文

　　　　実行文

行番号　最後の実行文

ex07-2.f

INTEGER TOTAL

TOTAL=0 !TOTALの初期値として0【この操作は重要：変数の初期化】

DO 10 I=1,10 !Iを1から10まで1ずつ増やしながら行番号10までを実行する命令

TOTAL=TOTAL+I

WRITE(\*,\*) 'I=',I,' TOTAL=',TOTAL

! WRITE(\*,'(A,I2,A,I3)') 'I=',I,' TOTAL=',TOTAL

10 CONTINUE ! DO 10 から　10 CONTINUEまでをDOブロックと呼ぶ

STOP

END

### 演習

ex07-2.fにおいて、

! WRITE(\*,'(A,I2,A,I3)') 'I=',I,' TOTAL=',TOTAL! DO文のブロック

の！を外し、次行のWRITE文をコメント行にして実行し、見やすさの点で考察せよ。

最後の実行文には’CONTINUE’を用いることが多い（基本的に何もしない実行文であり、主として行番号を付けるために用いられる）

上述までの例は、Iが1ずつ増える例を示したが、増分量（インクリメント：整数値）は1だけでなく、任意の整数値で良い。勿論、負の値でも良い。

ex07-3.f

INTEGER TOTAL

TOTAL=0

DO 10 I=0,10,2 !0から10まで2ずつIを増加させていく

TOTAL=TOTAL+I

WRITE(\*,'(A,I2,A,I3)') 'I=',I,' TOTAL=',TOTAL! DO文のブロック

10 CONTINUE

STOP

END

　また、初期値、終了値、インクリメントは整数変数を用いて表現することもできる。

ex07-4.f

INTEGER TOTAL

TOTAL=0

WRITE(\*,\*) '最初と終わりの数字を入力せよ!'

READ(\*,\*) M,N

DO 10 I=M,N

TOTAL=TOTAL+I

WRITE(\*,'(A,I2,A,I3)') 'I=',I,' TOTAL=',TOTAL

10 CONTINUE

STOP

END

## 7-2 DO文による二重ループ

ex07-5.f

INTEGER TOTAL

DO 10 **I**=0,10

**TOTAL=0**

DO 20 J=**I**,10

TOTAL=TOTAL+J

20 CONTINUE

WRITE(\*,'(A,I2,A,I3)') 'I=',I,' TOTAL=',TOTAL

10 CONTINUE

STOP

END

ex07-5.fの解説

1. はじめに、Iは0でJはI=0ではじまり、Jが10になるまでTOTAL=TOTAL+Jが繰り返される。これが終わると、DO 20のブロックから抜けだし、次行のWRITE文が実行される。
2. 次に、Iは２となり、JはI=1ではじまり、Jが10になるまでTOTAL=TOTAL+Jが繰り返される。ここで、注意しなければいけないのは、DO 20のブロックの前で、TOTAL=0としていることである。

したがって、

I=0のときはTOTAL=0+1+2+・・・・+10

I=1のときはTOTAL=1+2+・・・・+10

I=2のときはTOTAL=2+3・・・・+10

I=3のときはTOTAL=3+4+・・・・+10

・・・・

I=9のときはTOTAL=9+10

I=10のときはTOTAL=10

1. もし、このTOTAL=0がDO 10　のブロックの前であれば、TOTALの計算は

TOTAL=(0+1+2+・・・・+10) + (1+2+・・・・+10) + (2+3・・・・+10) +

・・・・ + (9+10) + (10)

となる。

二重ループの注意

① DOループの範囲は正しく指定しなければならない。下記のようなDO文の書き方は許されない。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | DO　10　・・・・・ |
|  |  |  |  |  |  | DO　20 ・・・・・  正しくない |
|  |  |  |  |  |  |  |
|  |  |  | 1 | 0 |  | CONTINUE |
|  |  |  | 2 | 0 |  | CONTINUE |
|  |  |  |  |  |  |  |

②端末文（DOブロックの最終行）を2つのDO文が指定している下記のような場合は、

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | **DO　10　・・・・・** |
|  |  |  |  |  |  | **DO　10 ・・・・・** |
|  |  |  |  |  |  |  |
|  |  |  | 1 | 0 |  | **CONTINUE** |
|  |  |  |  |  |  |  |

は

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | **DO　10　・・・・・** |
|  |  |  |  |  |  | **DO　20 ・・・・・** |
|  |  |  |  |  |  |  |
|  |  |  | 2 | 0 |  | **CONTINUE** |
|  |  |  | 1 | 0 |  | **CONTINUE** |

と同じ意味である。

## 7-3 その他の形式

### 7-3-1　行番号の省略

ex07-6.f

DO I=1,3

WRITE(\*,\*) I

END DO

DO I=1,3

DO J=101,103

WRITE(\*,\*) I,J

END DO

END DO

STOP

END

### 7-3-2　整数だけでなく実数でもできる

ex07-7.f

REAL\*8 D

DO A=1.,2.,0.2

WRITE(\*,\*) A !Aは単精度なので、精度が少し悪いことがわかる。

END DO

WRITE(\*,\*)

DO D=1.D0,2.D0,0.2D0

WRITE(\*,\*) D !Dは倍精度なので、精度があがっていることがわかる。

END DO !ただし、Dは1.8まで。これは、Dが2.D0となるときにわずかに、2より大きくなり、DOブロックの中に入れなかったためである。

WRITE(\*,\*)

DO D=1.D0,2.1D0,0.2D0

WRITE(\*,\*) D !上記の問題を解決するために、終わりの値に、刻み幅より小さい値（※）を加えればよい。

END DO

STOP

END

※DO　D=D1,D2+DX\*0.5d0,DXとしておけば、D2までの計算をしてくれる。

## レポート7

二次曲線y=x\*\*2の関数を考える。この関数の区間０から10までの積分値を近似的に求めるプログラム（report07.f）を作成せよ。ただし、刻み幅dxは0.1,0.01,0.001としたときの3ケースで計算せよ。計算は倍精度実数で行うこと。また、解析値（ANS=）との比較（刻み幅ごとに、近似値、近似値／理論値×100（％）を出力）、考察を行え。

ヒント：幅DXの短冊形に数値積分を行い、それを足しあわせていくことを考える。図１は与えられた関数である。ここで、DX=0.1d0の場合を考える。今、簡単のためy=f(x)と表記する。図１，２を参考にすると、近似値SUMは図中の短冊の面積を足しあわせることにより、求めることができる。

SUM=f(0)\*DX+f(0.1)\*DX+f(0.2)\*DX+f(0.3)\*DX＋・・・+f(9.7)\*DX+f(9.8)\*DX+f(9.9)\*DX

これをDOループを使って実行すればよい。少しだけ、ヒントを書く。

DX=0.1d0

SUM=0.d0

DO X=0.d0, 10.d0-DX+DX\*0.5d0, DX

SUM=???\*DX+???

END DO

補足説明：赤枠の10.d0-DX+DX\*0.5d0について。図３を見れば、Xは10.d0-DX（=9.9d0）まで求めればよいことがわかる。ただし、10.d0-DX（=9.9d0）とすると、X=9.9d0の手前の、X=9.8d0までしか計算してくれない可能性がある（ex07-7.fのときと同じ原因）。そのため、確実に10.d0-DX（=9.9d0）まで求めてくれるように、10.d0-DXの後ろにDX\*0.5d0を足す必要がある。上記の注釈（※）と同じこと。

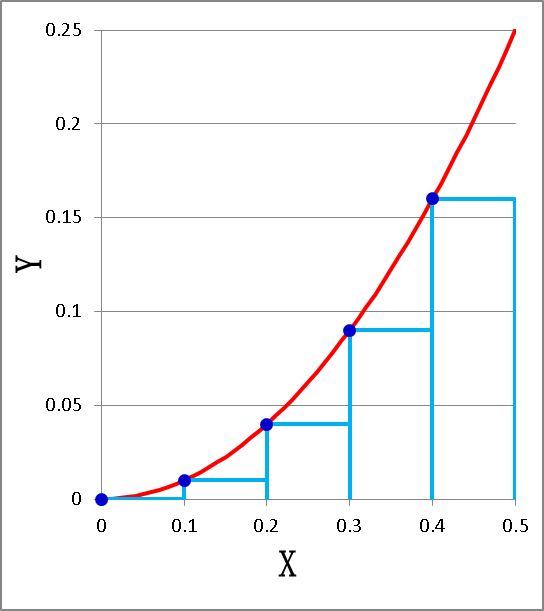


図１　　　　　　　　　　　　　図２　　　　　　　　　　　　　図３

# 第8回　配列（1次元）（整数、実数）

同じ型の複数の値を同じ名前の変数に保存して、添え字（インデックス）で識別するものを配列と呼びます。配列を理解することにより、プログラム作成が非常に楽になりますので、しっかり勉強してください。

1次元配列

|  |  |  |
| --- | --- | --- |
| 5.1 | 1番目の要素 | A(1) |
| 2.3 | 2番目の要素 | A(2) |
| 4.6 | 3番目の要素 | A(3) |
|  |  |  |
| 3.8 | n番目の要素 | A(n)  B(n,m)  B(2,3) |

　2次元配列 ・・・変数名（行番号,列番号）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5.1 | 3.6 | 1.8 |  | 5.2 | 1行目 |
| 2.3 | 2.4 | -3.2 |  | 8.1 | 2行目 |
| 4.6 | 5.2 | 4.2 |  | 6.2 | 3行目 |
|  |  |  |  |  |  |
| 3.8 | 1.4 | 3.8 |  | 5.4 | n行目 |
| 1列目 | 2列目 | 3列目 |  | m列目 |  |

　3次元、4次元と配列を作成することができる。

## 8-1 配列の宣言

　配列を宣言する場合には、変数名の後に要素数を「（　）」で囲んで指定します。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGR N(5) |
|  |  |  |  |  |  | REAL A(5),B (5,5) |

　添え字（インデックス）は、下記の通り、負、ゼロも使用できる。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | INTEGR N(0:5) |
|  |  |  |  |  |  | REAL A1(0:5),A2(-5:5),B1(0:5,0:5),B2(-5:5,-5:5) |

　宣言した配列の要素数を超えてその配列を用いることはできません。

ex08-1.f

INTEGER N(5)

REAL A(5)

DO I=1,5

N(I)=I\*2

A(I)=1./REAL(I)

END DO

DO I=1,5

WRITE(\*,\*) 'N(',I,')=',N(I)

END DO

DO I=1,5

WRITE(\*,\*) 'A(',I,')=',A(I)

END DO

STOP

END

ex08-2.f

REAL B(-5:0)

DO I=-5,0

B(I)=I

END DO

DO I=-5,0

WRITE(\*,\*) 'B(',I,')=',B(I)

END DO

STOP

END

ex08-3.f

REAL C(2,2)

DO I=1,2

DO J=1,2

C(I,J)=I+J

END DO

END DO

DO I=1,2

DO J=1,2

WRITE(\*,\*) 'C(',I,',',J,')=',C(I,J)

END DO

END DO

STOP

END

ex08-4.f

REAL D(-1:1,-1:1)

DO I=-1,1

DO J=-1,1

D(I,J)=I+J

END DO

END DO

DO I=-1,1

DO J=-1,1

WRITE(\*,\*) 'D(',I,',',J,')=',D(I,J)

END DO

END DO

STOP

END

## 8-2 配列の計算

ex08-5.f

REAL A(3),B(3)

DO I=1,3

A(I)=REAL(I)

B(I)=1./REAL(I)

END DO

SUM=0.

DO I=1,3

SUM=A(I)\*B(I)+SUM

END DO

WRITE(\*,\*) 'SUM=',SUM

STOP

END

## 8-3 文字列の配列

ex08-6.f

CHARACTER NAME(3)\*10

NAME(1)='SHIBAURA'

NAME(2)='TOYOSU'

NAME(3)='OHMIYA'

DO I=1,3

! WRITE(\*,\*) NAME(I)

WRITE(\*,'(A)') NAME(I)

END DO

STOP

END

ex08-7.f

CHARACTER NAME(3)\*10

NAME(1)='SHIBAURA'

NAME(2)='TOYOSU'

NAME(3)='OHMIYA'

DO I=1,3

! WRITE(\*,\*) NAME(I)(2:3)

WRITE(\*,'(A)') NAME(I)(2:3)

END DO

STOP

END

## レポート8

一次元配列A,Bの大きさは５とする。AはA(1)から順に1,3,5,7,9、BはB(1)から順に1,4,7,10,13が入っているとする。内積A・Bの値を求るプログラム(report08.f)を作成せよ。

# 第9回 配列（その２）

## 9-1　2×2の行列式を求めるプログラム

ex09-1.f

REAL A(2,2)

A(1,1)=1.

A(1,2)=2.

A(2,1)=3.

A(2,2)=4.

DET=A(1,1)\*A(2,2)-A(1,2)\*A(2,1)

WRITE(\*,\*) 'A='

DO I=1,2

WRITE(\*,'(2F5.1)') A(I,1),A(I,2)

END DO

WRITE(\*,\*) 'DET(A)=',DET

STOP

END

ex09-2.f

REAL A(2,2)

A(1,1)=1.

A(1,2)=2.

A(2,1)=3.

A(2,2)=4.

DET=A(1,1)\*A(2,2)-A(1,2)\*A(2,1)

WRITE(\*,\*) 'A='

DO I=1,2

WRITE(\*,'(2F5.1)') **(**A(I,J)**,J=1,2)** !出力の工夫

END DO

WRITE(\*,\*) 'DET(A)=',DET

STOP

END

## 9-2 　2×2の逆行列を求めるプログラム

ex09-3.f

REAL\*8 A(2,2),INV\_A(2,2),DET

A(1,1)=1.D0

A(1,2)=2.D0

A(2,1)=3.D0

A(2,2)=4.D0

DET=A(1,1)\*A(2,2)-A(1,2)\*A(2,1)

WRITE(\*,\*) 'A='

DO I=1,2

WRITE(\*,'(2F5.1)') (A(I,J),J=1,2)

END DO

WRITE(\*,\*) 'DET(A)=',DET

WRITE(\*,\*)

INV\_A(1,1)=A(2,2)/DET

INV\_A(2,2)=A(1,1)/DET

INV\_A(1,2)=-A(1,2)/DET

INV\_A(2,1)=-A(2,1)/DET

WRITE(\*,\*) 'INV\_A='

DO I=1,2

WRITE(\*,'(2F8.2)') (INV\_A(I,J),J=1,2)

END DO

STOP

END

ex09-4.f

REAL\*8 A(2,2),INV\_A(2,2),B(2,2)

! B=A INV\_A

A(1,1)=1.D0

A(1,2)=2.D0

A(2,1)=3.D0

A(2,2)=4.D0

INV\_A(1,1)=-2.0D0

INV\_A(1,2)= 1.0D0

INV\_A(2,1)= 1.5D0

INV\_A(2,2)=-0.5D0

!-----2x2の行列同士の積（始）--------------- ↓↓↓ここの部分が重要、だけど、理解が難しいところ！

DO I=1,2

DO J=1,2

**B(I,J)=0.D0**

DO K=1,2

B(I,J)=A(I,K)\*INV\_A(K,J)+B(I,J)

END DO

END DO

END DO

!-----2x2の行列同士の積（終）---------------　↑↑↑このの部分が重要、だけど、理解が難しいところ！

WRITE(\*,\*) 'B='

DO I=1,2

WRITE(\*,'(2F8.2)') (B(I,J),J=1,2)

END DO

STOP

END

補足

N×Nの行列A、B、Cがあり、C=ABのとき、

CのI行、J列の要素値C(I,J)は

C(I,J)=A(I,1)B(1,J)+A(I,2)B(2,J)+ A(I,2)B(2,J)+・・・+A(I,N-1)B(N-1,J)+ A(I, N)B(N,J)

この計算の部分は、ex09-4.fの

B(I,J)=0.D0

DO K=1,2

B(I,J)=A(I,K)\*INV\_A(K,J)+B(I,J)

END DO

に対応する。

## レポート9

A=, B=

とする。AB＝Cを求めるプログラム（report09.f）を作成せよ。

# 第10回 IF文

## 10-1 IF文

　何らかの条件を設定して、特定の文（実行文）を実行するかどうかを判断する。記述方法は以下のものがあります。

|  |
| --- |
| IF　(条件式)　実行文 |

条件式が真(True)ならば、実行文を実行し、偽（False）ならば実行しない。この書式では実行文は1行のみである。

良く使われる条件式として、2つの値の大小関係を条件とする場合がある。この書式として、

|  |  |  |
| --- | --- | --- |
| M＞N | IF (M .GT. N) | Greater Than |
| M≧N | IF (M .GE. N) | Greater or Equal |
| M＝N | IF (M .EQ. N) | EQual |
| M≦N | IF (M .LE. N) | Less or Equal |
| M＜N | IF (M .LT. N) | Less Than |
| MN | IF (M .NE. N) | Not Equal |

ex10-1.f

CHARACTER S\*1

N=1

IF(N.EQ.1) WRITE(\*,\*) N

WRITE(\*,\*)

IF(1.EQ.N) WRITE(\*,\*) N ! 変数を.EQ.のどちら側でもよい

WRITE(\*,\*)

S='Y'

IF(S.EQ.'Y') WRITE(\*,\*) 'YES'

STOP

END

条件式は複数でも使える。

|  |  |
| --- | --- |
| IF()のカッコ内の中身 | 意味 |
| ( a.eq.b .or. a.eq.c) | a＝bまたはa＝c |
| ( a.le.b .and. a.gt.c) | c < a ≦ b |
| (( a.le.b .and. a.gt.c) .or. a.le.d) | (c < a <= b)またはa ≦ d |

ex10-2.f

DO A=-1.,10.

IF( A.LE.5.0 .AND. A.GT.1.D0) WRITE(\*,\*) A

END DO

STOP

END

## 10-2 GOTO文

強制的に行番号のところへジャンプできる。

|  |
| --- |
| GOTO　行番号 |

ex10-3.f

!234567

NCOUNT=1

100 WRITE(\*,\*) 'COUNT=',NCOUNT

IF(NCOUNT.EQ.10) GOTO 999

NCOUNT=NCOUNT+1

GOTO 100

999 STOP

END

## 10-3 IFブロック

|  |
| --- |
| IF　(条件式)　THEN  　実行文1  実行文２  END IF |

条件式が真(True)ならば、実行文１，２を実行し、偽（False）ならば実行しない。IF文のブロックの最後には必ず END IFを書かなければいけない。

|  |
| --- |
| IF(条件式) THEN  実行文１  　 実行文２  ELSE  実行文3  　 実行文4  END IF |

条件式が真(True)ならば、実行文１，２を実行し、偽（False）ならば実行文３，４を実行する。

|  |
| --- |
| IF(条件式１) THEN  実行文１  　 実行文２  ELSE IF(条件式２) THEN  実行文3  　 実行文4  ELSE IF(条件式3) THEN  実行文5  　 実行文6  ELSE !ELSE文を加えることもできるし、省略もできる。  実行文７  　　実行文８  END IF |

ex10-4.f

WRITE(\*,\*) '試験の点数を入れてください!'

READ(\*,\*) A

IF(A.LT.60.) THEN

WRITE(\*,\*) 'D'

ELSE IF(A.LT.70.) THEN

WRITE(\*,\*) 'C'

ELSE IF(A.LT.80.) THEN !最初に条件を満たしたところのブロックに入る．

WRITE(\*,\*) 'B' !このブロックが実行されて，次に，END IF以下の実行文が実行される．

ELSE IF(A.LT.90.) THEN

WRITE(\*,\*) 'A'

ELSE IF(A.LE.100.) THEN

WRITE(\*,\*) 'S'

ELSE

WRITE(\*,\*) 'すごい！あなたの点数は100点を超えています！'

END IF

STOP

END

ex10-5.f

100 WRITE(\*,\*) '試験の点数を入れてください!'

READ(\*,\*) A

IF(A.LT.60.) THEN

WRITE(\*,\*) 'D'

ELSE IF(A.LT.70.) THEN

WRITE(\*,\*) 'C'

ELSE IF(A.LT.80.) THEN !最初に条件を満たしたところのブロックに入る．

WRITE(\*,\*) 'B' !このブロックが実行されて，次に，END IF以下の実行文が実行される．

ELSE IF(A.LT.90.) THEN

WRITE(\*,\*) 'A'

ELSE IF(A.LE.100.) THEN

WRITE(\*,\*) 'S'

ELSE

WRITE(\*,\*) 'すごい！あなたの点数は100点を超えています！'

WRITE(\*,\*) 'このプログラムを終了します。'

! STOP

GOTO 999

END IF

GOTO 100

999 STOP

END

※IFブロックは入れ子構造にすることもできる。

|  |
| --- |
| IF(条件式１) THEN  実行文１  　 実行文２  ELSE IF(条件式２) THEN  IF(条件式①) THEN  実行文3①  　 実行文4①  ELSE IF (条件式②) THEN  実行文3②  　 実行文4②  END IF  ELSE IF(条件式3) THEN  実行文5  　 実行文6  END IF |

## レポート10

地盤材料は粒径D(mm)により、下記のように分類される。

粘土 ：　　　 D＜0.005

シルト： 0.005≦D＜0.075

砂 ： 0.075≦D＜ 2.0

礫 ： 2.0 ≦D＜75.0

石 ：75.0 ≦D

粒径(mm)をキーボードで入力すると、土質名が出力されるプログラムreport10.fを作成せよ。

# 第11回　ファイル操作

**ex11-1a.f**

!　学習目的：ファイルのopen, close

open(10,file='ex11-1.txt')

write(10,\*) 'abcd'

do i=1,5

write(10,\*) i

end do

close(10)

stop

end

**ex11-1b.f**

!　学習目的：ファイルのopen, close

character str\*4

open(20,file='ex11-1.txt')

read(20,\*) str

write(\*,\*) str

do i=1,5

read(20,\*) j

write(\*,\*) j

end do

close(20)

stop

end

**ex11-2a.f**

!　学習目的：open文のオプション：status

character str\*4

open(20,file='ex11-1.txt',status='old') !存在していなければ実行時エラーとなる

read(20,\*) str !全角１文字は半角2文字分

write(\*,\*) str

do i=1,5

read(20,\*) j

write(\*,\*) j

end do

close(20)

stop

end

**ex11-2b.f**

!　学習目的：open文のオプション：status

character str\*4

open(10,file='ex11-1.txt',status='new') !※

write(10,\*) 'abc'

close(10)

stop

end

!※ ex11-1.txtは既に存在しているので，実行時にエラーになる．

!　 既存のファイルが存在する場合，これをoverwriteしたくないときに使う．

**ex11-3.f**

!　学習目的：err=,end=,rewind,backspace

character str\*4

open(10,file='ex11-3.txt')

write(10,\*) 'abcd'

do i=1,5

write(10,\*) i

end do

close(10) !ひとつのファイルに対し、書き込みと読み込みを同時にすることはできないので、いったん閉じる必要がある。

open(10,file='ex11-3.txt')

read(10,\*,err=20) m

! read(10,\*) m !上記のread文の先頭に!を付けてコメント文にし、この行の!を外して実行してみる。

20 write(\*,\*) 'reading err !'

pause !続行する場合にはgoとタイプする。

rewind(10) !ファイルの先頭に戻る

read(10,\*) str

write(\*,\*) str

read(10,\*) m

write(\*,\*) m

backspace(10) !read()を実行すると，読込む行が次に行くが，backspaceをすることにより１行戻る

read(10,\*) m

write(\*,\*) m

pause

rewind(10)

read(10,\*) str

write(\*,\*) str

30 read(10,\*,end=999) m !ファイルに最後にたどり着くと行番号999へ飛ぶ

write(\*,\*) m

goto 30

999 close(10)

stop

end

**ex11-4.f**

!　学習目的：書式付のファイルの書き込み，読込み

implicit real\*8 (a-h,o-z)

open(10,file='ex11-4.txt')

do x=0.d0,1.d0,0.1d0

write(10,'(a,f5.2,a,f10.7)') 'x=',x,' x^2=',x\*x

end do

close(10)

open(20,file='ex11-4.txt')

10 read(20,'(3x,f5.0,5x,f10.0)',end=999) x,x2 !読み込むときは，f5.0とカラム数があっていればよい

write(\*,\*) x,x2

goto 10

999 close(20)

stop

end

**ex11-5.f**

!　学習目的：書式なしのファイルの読込み

open(10,file='ex11-5.txt')

write(10,\*) '1,2,3' !数字の区切りは，カンマかスペース

write(10,\*) '1 2 3' !数字の区切りは，カンマかスペース

close(10)

open(20,file='ex11-5.txt')

m=0

10 read(20,\*,end=999) i,j,k !数字の区切りは，カンマかスペース

write(\*,\*) i,j,k

m=m+1 !ファイルの行数をカウント

goto 10

999 close(20)

write(\*,\*) 'ex11-5.txtの行数=',m

stop

end

**ex11-6.f**

!csv形式で出力し、EXCELで読み込む

PI=4.\*ATAN(1.0)

DTHETA=PI/100.

OPEN(10,FILE='EN.CSV')

DO THETA=0.0, 2.\*PI, DTHETA

X=COS(THETA)

Y=SIN(THETA)

WRITE(10,'(F7.3,A,F7.3)') X,',',Y

END DO

CLOSE(10)

STOP

END

!出力ファイルEN.CSVをダブルクリックして、EXCELで開かれることを確認せよ。

!次に、X,Yデータのグラフを作成せよ。

**ex11-7.f**（これは、おまけなので、スキップしてよい）

!　学習目的：バイナリｰファイル(form='unformatted')

implicit real\*8 (a-h,o-z)

open(10,file='ex11-7.txt',form='unformatted') !拡張子は便宜上txtとしているが、テキストファイルではない

do x=0.d0,1.d0,0.1d0

write(10) x,x\*\*2

end do

close(10)

open(10,file='ex11-7.txt',form='unformatted')

10 read(10,end=999) x,x2 !書き込み時の書式を把握している必要がある

write(\*,\*) x,x2

goto 10

999 close(10)

stop

end

補足：構造計算などの汎用ソフトのそろばんの部分はフォートランで書かれていることも多い。通常、計算結果はバイナリーファイルで出力され、後から、このバイナリーファイルを読み込み、指定する箇所の結果をテキストファイルで出力する。

## レポート11

x=2cos(θ), y=3sin(θ)、θ=0～２πのx、yをCSV形式（カンマ付きファイル）で出力（report11.csv）するプログラム（report11.f）を作成せよ。また、エクセルでグラフを作成し、これも提出せよ。

# 第12回　SUBROUTINE

**ex12-1.f**

ポイント

・サブルーチンはプログラムの中で繰り返し実行される計算を一つのプログラムとして実行するものである。

・例えばex12-1.fのSUBROUTINE WAはA,Bの値をメイン文から受け取り、AとBの和をCに代入して、また、A,B,Cの値をメイン文に戻す内容となる。

・SUBROUTINEはCALL文で呼び出される。

・SUBROUTINEの最後は、

RETURN

END

で終わる。

!　学習目的：サブルーチン

A=2.

B=3.

CALL WA(A,B,C)

WRITE(\*,\*) C

WRITE(\*,\*)

CALL SEKI(A,B,C)

WRITE(\*,\*) C

WRITE(\*,\*)

STOP

END !ここまでの文をメイン文という

SUBROUTINE WA(A,B,C)

C=A+B

RETURN

END

SUBROUTINE SEKI(A,B,C)

C=A\*B

RETURN

END

!その他：メイン文での実行文は少なくし，計算はサブルーチンで行うような構成にする

!とプログラムは読みやすくなる．

**ex12-2.f**

!　学習目的：サブルーチン(引数の型の対応)

INTEGER A,B,C

A=1

B=2

CALL WA(A,B,C) !（）内を引数と呼ぶ

WRITE(\*,\*) C

STOP

END

SUBROUTINE WA(A2,B2,C2) !変数名は異なっていてもよい

INTEGER A2,B2,C2 !ただし，引数の型はCALL文の中と対応していなければいけない

C2=A2+B2

RETURN

END

**ex12-3.f**

!　学習目的：サブルーチン(入れ子構造)

A=2.

CALL SUB1(A,B,C) !（）内を引数と呼ぶ

WRITE(\*,\*) 'A=', A

WRITE(\*,\*) 'B=', B

WRITE(\*,\*) 'C=', C

STOP

END

SUBROUTINE SUB1(A,B,C)

A=2.\*A !A=4. サブルーチンの中で変数の値を変えることができる．

CALL SUB2(A,B) !サブルーチンの中からサブルーチンをCALLできる

C=A+B+1 !C=4.+5.+1.=10.

RETURN

END

SUBROUTINE SUB2(A,B)

B=A+1. !B=5. !◆２

RETURN

END

!引数には，CALLする時点で値をもってSUBROUTINE文に行くものと，

!CALLする時点では値を持たず，SUBROUTINEの中で値を得て戻ってくるものの２種類がある。

**ex12-4.f**

!　学習目的：サブルーチン(配列:その１)

DIMENSION A(5),B(5)

DO I=1,5

A(I)=I

END DO

CALL SUB1(A,B) !配列全体をSUB1に渡す場合

DO I=1,5

WRITE(\*,\*) B(I)

END DO

WRITE(\*,\*)

DO I=1,5

CALL SUB2(A(I),C) !配列の要素のみ渡す場合

WRITE(\*,\*) C

END DO

STOP

END

SUBROUTINE SUB1(A,B)

DIMENSION A(5),B(5)

DO I=1,5

B(I)=A(I)

END DO

RETURN

END

SUBROUTINE SUB2(A,B) !Aは配列ではない

B=A

RETURN

END

**ex12-5.f**

!　学習目的：サブルーチン(配列:その２)

PARAMETER (ND=5)

DIMENSION A(ND),B(ND) !◆１

DO I=1,5

A(I)=I

END DO

CALL SUB1(A,B,ND)

DO I=1,5

WRITE(\*,\*) B(I),' IN MAIN'

END DO

STOP

END

SUBROUTINE SUB1(A,B,ND)

DIMENSION A(ND),B(ND),C(ND\*2) !配列の大きさの指定のとき演算も使える

DO I=1,5 !A,Bはグローバルな変数であるが，Cはローカルな変数

B(I)=A(I)

END DO

CALL SUB2(A,C,ND)

DO I=1,5

WRITE(\*,\*) C(I),' IN SUB1'

END DO

RETURN

END

SUBROUTINE SUB2(A,C,ND)

DIMENSION A(ND),C(ND\*2)

DO I=1,5

C(I)=2\*A(I)

END DO

RETURN

END

!◆１：配列の大きさをPARAMETERで指定し，CALLするときに渡すのを薦める．

!　　　こうすることによって，配列の大きさの管理が容易になる．

**ex12-6.f**

!　学習目的：COMMON文

PARAMETER (ND=5)

CHARACTER\*2 S1,S2

COMMON /ABC/A(ND),B(ND),C,N,M/STR/S1,S2 !COMMON文で配列の宣言もできる．

!/ブロック名１/変数１/ブロック名２/変数２

!注意：文字型の変数だけは，他の型のブロックに入れることができない．

DO I=1,ND

A(I)=I

END DO

CALL SUB1 !引数でなく、COMMON文だけでデータを渡すこともできる。

DO I=1,ND

WRITE(\*,\*) B(I),S1,S2

END DO

STOP

END

SUBROUTINE SUB1

PARAMETER (ND=5)

CHARACTER\*2 S1,S2

COMMON /ABC/A(ND),B(ND),C,N,M/STR/S1,S2

DO I=1,ND

B(I)=2\*A(I)

END DO

S1='AB'

S2='CD'

RETURN

END

**ex12-7.f**

!　学習目的：サブルーチン側で配列の大きさ，文字数にワイルドカード(\*)を使用

CHARACTER\*10 S1

DIMENSION A(2),B(2,2)

CALL SUB1(S1,A,B)

WRITE(\*,\*) S1

WRITE(\*,\*) A(1)

WRITE(\*,\*) A(2)

WRITE(\*,\*) B(1,1)

WRITE(\*,\*) B(1,2)

WRITE(\*,\*) B(2,1)

WRITE(\*,\*) B(2,2)

STOP

END

SUBROUTINE SUB1(S1,A,B)

CHARACTER S1\*(\*) !文字数にワイルドカードが使用可

DIMENSION A(\*),B(2,\*) !1次元配列でもワイルドカード使用可

!ただし，２次元の場合，行数

はきちんと指定しないといけない．

S1='ABC'

A(1)=1.

A(2)=2.

B(1,1)=3.

B(1,2)=4.

B(2,1)=5.

B(2,2)=6.

RETURN

END

## レポート12

関数exp(x)は、テーラー展開を使うと次式のように近似できる。

x=0～１は0.1刻みで、x=1～10は、１刻みで組み込み関数のdexp(x)と上記の近似式をサブルーチンで計算した結果とを比較するプログラム(report12.f)を作成せよ。ただし、倍精度で計算すること。

条件：

1. 出力は以下を使用すること。

WRITE(\*,'(F5.1,2E18.10,F10.5)') X,EXPX,DEXP(X),DEXPX/EXP(X)\*100.D0

1. サブルーチンのコールは以下のようにすること。

CALL SUB\_EXP(X,EXPX)

補足：組み込み関数exp(x)は単精度の計算値、dexp(x)は倍精度の計算値

# 第13回　最終レポート課題（レポート１３）