Introduction to
Scientific & Engineering Computing
BIL 102FE (Fortran) Course
for
Week 3

Dr. Ali Can Takinacı

Assistant Professor

in

The Faculty of Naval Architecture and Ocean Engineering

80626

Maslak – Istanbul – Turkey

DATA HANDLING

Integer, Real, Character String and Named constants
in F

Two fundamental types of numbers

Real Numbers : have fractional part
Integers : have not fractional part

Integer: whole number, held exactly in the computer's memory. It has a limited range (-2×10^9 and $+2\times10^9$ on a typical 32-bit computer)

Real Number: stored as a floating-point number. It is always held an approximation with a fixed number of significant digits. Large range (between about -10^{38} and $+10^{38}$ to 7 or 8 significant digits on the same 32-bit computer).

Real and integer variables

The integer variables are declared by a statement => integer :: a, b, c

a, b and c numbers may be varied from 99999999 to –99999999 depending on the compiler

real variables are declared as => real :: x, y, z
stored with integer part + a string of digits

100 000 000.0 can not be stored. It needs 9 digits before decimal point.
so it would be written 0.1×10^{9}

0.000 000 004 would be the same as $4 \div 10^{8}$ or 0.4×10^{-9}

it must be remembered that the real arithmetic is always an approximation.

Arithmetic expressions and assignments

assignment or by a read statement
are
two ways in which a variable can be given a value during the execution of a program

Assignment

name = expression

current value of expression is assigned to the variable name

a = b + c

current value of b is added to current value of c and the result of the sum is assigned and stored to the variable a.

Expression

$a = b + c$

a, b and c are real variables

b = 2.80 c = 3.72 => a = 6.52 or (~6.5199999)

a, b and c integers

b = 17 c=391 => 408 (exact result - no approximation)

a is integer and b, c are real

result is truncated by throwing away the fractional part

b = 2.80 c = 3.72 => a = 6 (exact number)

Integers are converted to real without losing any accuracy

Integer 12345678 - Real 12345678.0 or 1.234568×10^7

Arithmetic Expressions

the five primary arithmetic operations

addition (+)

subtraction (-)

multiplication (*)

division (/)

exponentiation (**)

Priorities

** => high

* and / => medium

+ and - => low

As an example the expression

$a = b + c*d/e-f**g/h + i*j + k$ – $a = b + c \times \frac{d}{e} - \frac{f^{g}}{h} + i \times j + k$ (formula representation)

The evaluation proceeds as follows

1) Calculate f**g and save it in temp_1
2) Calculate c*d " temp_2
3) Calculate temp_2/e " temp_3
4) Calculate temp_1/h " temp_4
5) Calculate i*j " temp_5
6) Calculate b+temp_3 " temp_6
7) Calculate temp_6-temp_4 " temp_7
8) Calculate temp_7+temp_5 " temp_8
9) Calculate temp_8+k and store it in a

programmer can not access to temporary variables
Principle is simple
namely
each calculation step consists of one operator having two operands

In a mixed-mode expression (mixed with integer and real numbers), the integer value is converted to real

a =b*c/d

b=100.0 (real) c=9, d=10 (integers)

1) b*c is evaluated. c is converted to real (9.0) => 900.0

2) d is converted to real (10.0) => 900.0/10.0 = 90.0

a =c/d*b (mathematically equivalent but !!!)

b=100.0 (real) c=9, d=10 (integers)

1) c/d is evaluated. Both are integer (no fractional part)
so 9/10=0.9 is truncated to 0

2) Result is zero --- too late
(integer division phenomenon)

use of () can alter the order of evaluation

w=x*(z-y)

1) (z-y) is first evaluated

2) Result is multiplied by x and result is assigned to w

```
a = b + c*d/e – (f**g)/h + i*j + k
```

the spaces are for purely for the human reader
they are ignored by the F compiler

Constants

The variable that its value could not be changed

it is defined as a constant

real, integer or exponential form

123 => Integer constant
1.23 => Real constant
1.e-6 =>Exponential constant

Exponential form

me exp

me => mantissa exp=>exponent

the value 0.000001 or 10^{-6} may be written

1e-6

100e-8

0.1e-5

List-directed input and output of numeric data

read *, var_1,var_2,.....
print *, item_1,item2,.....

- They both have an almost identical syntax.
- The read statement may only contain variable names
- Print statement may also contain constants or expressions
- These lists of names or other items are referred to as input-list or output-list
- The asterisk following the read or print indicates that list-directed formatting is to take place.

Handling character data

The declaration statement is

`character (len=`*`length`*`) :: name1, name2`

This declares one or more character variables, each of which has a length of *length*

In the following program
the three variable string_1, string_2 and string_3 are stored

```
program character_example
character(len=3) :: string_1
character(len=4) :: string_2,string_3
string_1="End"
string_2=string_1
string_3="Final”
print*, string_1,string_2,string_3
end program character_example
```

output of prog.

```
EndEnd Fina
```

The combination of two or more strings to form a third is known as the concatenation procedure, and this is carried out by the concatenation operator, which is the only operator provided in F,_consisting of two consecutive slashes.

char=”Fred”//” and “//”Wilma”

Named constants

The constants, which are no need to be altered, are called named constants. They are used by the parameter attribute in a declaration statement.

```
type, parameter :: name1=constant_expression_1, ...
```

for example

```
real, parameter :: pi=3.1415927, pi_by_2=pi/2.0
```

The use of named constant is purely to make the program easier to read and to avoid errors in typing long constants. For example the statement

```
area=3.1415927*r*r
```

can be written by using the named constant of “pi”

```
area=pi*r*r
```

the end