Lecture 2
FORTRAN Basics

Lubna Ahmed

Fortran basics

- Data types
- Constants
- Variables
- Identifiers
- Arithmetic expression
- Intrinsic functions
- Input-output

Program layout

PROGRAM program name

IMPLICIT NONE

Declaration statements(data defining)

Executable statements (data processing)

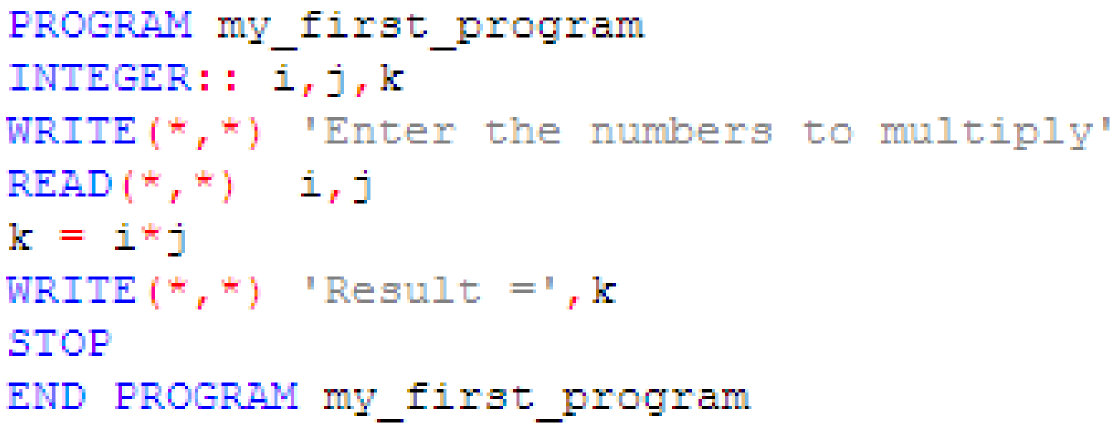
Data input

Calculation

Data output

END PROGRAM program name

Example



```
PROGRAM my_first_program
INTEGER:: i,j,k
WRITE(*,*) 'Enter the numbers to multiply'
READ(*,*)  i,j
k = i*j
WRITE(*,*) 'Result =',k
STOP
END PROGRAM my_first_program
```

Statements

- Form the basis of any Fortran program, and may contain from 0 to 132 characters.
- All statements begins with a keyword except assignment statement.
- INTEGER, REAL, LOGICAL, READ, WRITE, PRINT, OPEN etc are some keywords.
- Generally, there will be one statement per line. However, multiple statements may appear on a line if they are separated by semi-colons but not recommended.

```
A = 1; B = 1; C = 1
```

Comments

Any characters following an exclamation mark (!) are commentary, and are ignored by the compiler.

Significance of Blank

Blanks are generally not significant, i.e. you can use them to improve readability.

FORTRAN Alphabets

Fortran only uses the following character:

Letters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h I j k l m n o p q r s t u v w x y z

Digits: 0 1 2 3 4 5 6 7 8 9

Special Character:

space

‘ ‘’ () * + - / : = _

! & $; < > % ? , .

Continuation of line

If a statement is too long to fit on a line, it will be continued on the next line if the last non-blank character in it is an ampersand (&):

```
A = 174.6 *            &
&(T - 1981.2) ** 3
```

The above is equivalent to

A=174.6*(T-1981.2)3**

Data types

Data type	Characteristic
Integer	Whole numbers stored exactly
Real	Numbers, which may have fractional parts, with limited precision
Double precision	Similar to real but with greater precision
Complex	Stored as an ordered pair of real numbers
Logical	A Boolean value: either true or false
Character	A string of characters of fixed length

Data types

The data that are handled by a program fall into two main types:

- **Constants:** Constants do not change during the execution of FORTRAN program.
- **Variables :** in which the program will store its input, output, constants and intermediate results, may change value during execution of program.

- There are five intrinsic or built-in types of FORTRAN constants and variables. Three of them are of **numeric** (types INTEGER, REAL and COMPLEX),one is **logical**(type logical) and one consists of **string of character**(type CHARACTER).

Constants

Constant are the token used to denote the value of a particular type. Values of constants do not change during program execution.

Integer constant

- An integer constant is any number that does not contain a decimal point.
- The general form is a sign (plus or minus) followed by a string of one or more digits. All other characters are prohibited. If the number is positive the plus sign is optional.

❖ Examples of valid integer constants:

- ❑ 0
- ❑ 137
- ❑ -2516
- ❑ +17

❖ Some invalid integers:

- ❑ 5,28,000 (embedded commas are illegal)
- ❑ 16.0 (If it has a decimal point, it is not an integer constant)
- ❑ - - 5

Constants cont'd

Real Constant

There are two representation

1. **Decimal representation:-** A decimal point must be present, but no commas are allowed, can have an optional sign.

Correct Examples :	23.45	.12	123.	-0.12	-.12
Incorrect Examples :	1,435.95	75	123.5-	$12.34	

2. **Exponential representation:-** It consists of an integer or a real number in decimal representation, followed by exponent.

Correct Examples:

- 12.3456E2 or 12.3456e2: equal to 1234.56
- -3.14e1 or -3.14e1 : equals to -31.4
- -1.2E-3 or -1.2e-3 : equals to -0.0012
- 12E3 or 12e3 : equals to 12000.0
- 0E0 or 0e0 : equals to 0.0

Incorrect Examples:

- 12.34E1.2 : the exponential part must be an integer constant
- 12.34-5 : there is no exponential sign E or e

Constants cont'd

Complex constants

A **complex constant** has the form of two real or integer constants separated by a comma enclosed in a pair of parentheses, e.g. (A,B) for A+iB.

Examples:

```
(3.14,-5.67)
(0,0)
(-0.999,2.718E15)
```

Logical constants

There are only two possible **logical constants**. They are expressed as .TRUE. and .FALSE.

Example: X.GT.Y is .TRUE. (if X is greater than Y)

X.GT.Y is .FALSE. (if X is not greater than Y)

Constants cont’d

<u>Character String</u>

A ***character constant*** consists of a string of characters enclosed in a pair of apostrophes which act as quotation marks.

. **Correct Examples:**

- ‘John’ or “John” : content = **John** and length = **4**
- ‘ ’ or “ ” : content = a single **space** and length = **1**
- ‘John Dow#2’ or “John Dow#2” : content = **John Dow#2** and length = **10**
- ‘’ or “” : content = **nothing** and length = **0** (empty string)

Incorrect Examples:

- ‘you and me : the closing apostrophes is missing.
- Hello, world’ : the opening apostrophes is missing.
- ‘Hi” or “Hi’ : the opening and closing quotes do not match.

Note: If single quote is used in a string, the double quotes should be used to enclosed the string: **“John’s pen”**

Implicit Type Rule

First letter of the name	Implicit type
A to H	Real
I to N	Integer
O to Z	Real

This (unhelpful) rule still applies in Fortran 90 by default ,to ensure compatibility of code written under earlier versions.

Example:- Interest rate = 0.12

program where Interest rate is not declared explicitly will assign the value 0 to the variable.

To guard against such errors it is strongly recommended that the statement **IMPLICIT NONE** be used at the start of all programs.

Variables

- In mathematics, a symbolic name is often used to refer a quantity. For example, the formula:

  $$A = l.w$$

  is used to calculate the are (A) of a rectangle with a given length (*l*) and a given width (*w*). These symbolic names, A, *l*, *w*, are called variables.
- And the same thing can be said about the variables in Fortran.
- The type of a FORTRAN variable must be one of the <u>six data types</u> mentioned before.
- The type of each variables determines the type of value that may be assigned to that variable.

Variables cont'd

- A variable is a memory location whose value may be changed during execution of a program. A variable's name is constructed in a similar way as the memory name.
- A variable has a **type** which determines the **type** of constant it may hold. It is given a **type** in a type declaration.

INTEGER X

REAL INTEREST

CHARACTER LETTER

REAL :: A = 1

Here, X, INTEREST, LETTER are undefined variable and A is defined variable.

Double colon (::) is used to initialize a variable.

Variables cont’d

```
program vertical
! Vertical motion under gravity
real :: g ! acceleration due to gravity
real :: s ! displacement
real :: t ! time
real :: u ! initial speed ( m / s)
! set values of variables
g = 9.8
t = 6.0
u = 60
! calculate displacement
s = u * t - g * (t**2) / 2
! output results
write(*,*) ’Time = ’,t,’ Displacement = ’,s
end program vertical
```

There are four variables g, s, t, u

Variables cont'd

Some rules for FORTRAN variable name:

1. **1 to 31 characters in length.**
2. **May use A-Z, a-z, 0-9 and space. Upper and lower cases are equivalent.** For example therefore the variables T and t, and the program names vertical and Vertical are identical.
3. **First character of a name must be alphabet A-Z, a-z.** For example: 3DYR is an incorrect variable.
4. Don't use keywords or library functions as a variable name.
5. If a variable name is undeclared in the program it is assumed a numerical variable by the compiler and if first character of the name is I, J, K, L, M or N, then the variable will be an integer variable and if the first character is any of A-H, O-Z then it will be real variable.
6. Always use meaningful variable.

Character Variable and Declaration

Since a character string has a length attribute, a length value must be attached to character variable declarations. There are two ways of to do this:

```
CHARACTER (LEN= 15) :: name, street
CHARACTER(15): : name, street
```

If a variable can only hold a single character, the length part can be removed.

```
CHARACTER(LEN =1)    :: letter, digit
CHARACTER(1)         :: letter, digit
CHARACTER             :: letter, digit
```

If you want to declare character variables of different length with a single statement, then

```
CHARACTER(LEN = 10)  :: city, nation*20, box, digit*1
```

If character variable is declared in the following way…

```
CHARACTER(LEN= *) :: city, nation
```

Then, the actual length of these character variables city and nation are unknown and will be determined elsewhere.

Parameter Attribute

In some cases, one wants to assign a name to a particular value.

For example: value of Pi = 3.1415926

It is tedious to write the value of Pi in every time in a program. To avoid this parameter attribute is used.

```
REAL, PARAMETER :: E=2.71828, PI = 3.1415926
CHARACTER (LEN = 4), PARAMETER :: Name = ‘John’, City = “LA”
```

Arithmetic Operators

Fortran has four types operators: Arithmetic, Rational, Logical, Character.

Types	Operator	Associability
Arithmetic	**	Right to left
	* /	Left to right
	+ -	Left to right
Rational	< <= > >= == /=	None
Logical	.NOT.	Right to left
	.AND.	Left to right
	.OR.	Left to right
	.EQV. .NEQV.	Left to right

Arithmetic Operators cont’d

- The order of evaluation of an expression:
  - Sub-expressions in parenthesis
  - Function reference
  - Exponentiation, i.e., raising to a power
  - Multiplication and division
  - Addition and subtraction
- Within each of these groups evaluation proceeds from left to right, except that exponentiations are evaluated from right to left. Therefore X**Y**Z is equivalent to X**(Y**Z)

Arithmetic Operators cont’d

Incorrect	Problem	Correct
A** -B	Consecutive operators	A**(-B)
A(B+ 3.6)	Missing operator	A*(B+ 3.6)
A* 2.75 – B*(C+ D))	Unpaired brackets	A* (2.75 – B*(C+ D))

Arithmetic Operators cont’d

Example

Variables a, b, c, d, e, f, and g have been initialized to the following values: a= 3. b= 2. c= 5. d=4. e=10. f=2. g=3.

Evaluate the following FORTRAN assignment statement:

Output = a* b+ c*d+ e /fg**

Solution:

- *Fill in the numbers* : output = 3.* 2. + 5. * 4. +10. / 2. ** 3.
- *Evaluate 2. ** 3.* : output = 3.* 2. + 5. * 4. +10. / 8.
- *Evaluate multiplications and divisions from left to right:*

Output = 6. + 5. *4. + 10. / 8.

Output = 6. + 20. + 1.25

- *Evaluate additions:* output = 27.25

Integer Division

- Integer division always produces a result which is another integer value.
- Any fractional part is truncated, i.e., rounded towards zero.

Example: 8/3 = 2

This makes it especially important to provide a decimal point at the end of a real constant even if the fractional part is zero.

Mixed Mode Expression

Fortran 90 allows operand in an expression to be of different type. The general rule is that the weaker or simpler type is converted into the stronger type. Since integer type is the simplest, this means that operations involving real and integer operand will be done in real arithmetic.

Example:

- 1.0/4 → 1.0/4.0 → 0.25
- 3.0+8/5 → 3.0+1 → 3.0+1.0 → 4.0
- 3+8.0/5 →3+8.0/5.0 →3+1.6 → 3.0+1.6 →4.6
- 3/4*5.0 → 0*5.0 →0
- 5.0*3/4 → 5.0*3.0/4 →15.0/4 → 15.0/4.0 →3.75

Arithmetic Intrinsic Function

Trigonometric functions:

SIN (x)	sine of x radians
COS (x)	cosine of x radians
TAN (x)	tangent of x radians
ASIN (x)	arc sine of x
COS (x)	arc cosine of x
ATAN (x)	arc tangent of x

Transcendental functions

SQRT (x)	square root
LOG (x)	natural logarithm
EXP (x)	returns the exponential
LOG10 (x)	logarithm to base 10

Others

MIN (x_1, x_2, …………)

MAX (x_1, x_2, ………..)

ABS (x)

MOD (x_1, x_2)

Input- Output

Fortran provides two types of input /output statements:

- Formatted input/output
- List-directed input/output

List-directed output

PRINT*, *output-list*

Or, WRITE(*,*), *output-list*

List-directed input

READ*, *input-list*

Input- Output

Example:-

```
PROGRAM INPUT_OUTPUT
IMPLICIT NONE
REAL X, Y           ,SUM     ! Variable declaration
WRITE(*,*), ‘GIVE THE VALUE OF X AND Y ’
READ*, X,Y
SUM = X+Y           ! Executable statement
WRIRE(*,*), ‘SUMMATION OF GIVEN NUMBER:’, SUM
END PROGRAM INPUT_OUTPUT
```

Assignment

1. An orifice meter is used to measure the flow rate in pipes. The flow rate is related to the pressure drop by an equation of the form:

where u = fluid velocity

Δp = pressure drop

ρ = density of the flowing fluid

c = constant of proportionality

Write a computer program that reads pressure drop and density and calculates the fluid velocity. A constant of proportionality must be defined and used also.