FORTRAN 90: Basic FORTRAN

Meteorology 227

Fall 2024

FORTRAN Data Types

- INTEGER
- REAL
- COMPLEX
- CHARACTER
- LOGICAL

INTEGER/REAL

- INTEGER
  - Whole numbers (positive, negative, or zero)
  - 0, 137, -2516, 17745 are valid integers
- REAL
  - Ordinary decimal notation or exponential notation.
  - 1.234, -0.01636, +56473.
  - 3.37456E2, 0.337456E3, 337.456E0,
  - 33745.6E-2, 337456E-3

Character Strings or Strings

- Sequences of symbols from the FORTRAN character set.
  - ANSI standard character set (Table 2.1)
- Must be enclosed between double quotes or between apostrophes (single quotes).
- Length = number of characters in string.
- “PDQ123-A” has a length=8
- “” has a length =
- ‘Don’t’ or “Don’t”

Identifiers

- Names used to IDENTIFY programs, constants, and variables.
- Must begin with a letter, which may be followed by up to 30 letters, digits, or underscores.
  - R2-D2 ?
  - 6Feet ?
- Use meaningful identifiers that suggest what they represent.
- FORTRAN 90 is not case-sensitive

Variables

- Associated with memory locations.
- Variable names are identifiers and must follow the rules for forming valid identifiers.
- Type statements
  - The type of a FORTRAN variable determines the type of value that may be assigned to the variable.
- Examples
  - INTEGER :: Hours
  - REAL :: Temp
  - INTEGER :: Hour, Minute, Second
  - REAL :: Temp, Dew_Point, Wet_Bulb

Variables cont.

- More examples
  - CHARACTER(LEN=20) :: Name
  - CHARACTER(20) :: Name
  - CHARACTER :: First_Initial
- Naming Cautions
  - Any variable whose type is not explicitly declared in a type statement is subject to implicit naming conventions.
    - I,J,K,L,M,N → INTEGER
    - All others → REAL

IMPLICIT NONE

- Implicit naming can cause problems!
  - Mass = 12.345

- IMPLICIT NONE
  - Should be used in every program (and module).
  - All variables and constants must be specified explicitly.

Variable Initialization/Parameters

- Initialization
  - REAL :: Temp = 28.5, Dew_Point = 26.5
  - REAL :: Temp = 28.5
  - REAL :: Dew_Point = 26.5
- Parameters
  - Type-specifier, PARAMETER :: List
  - INTEGER, PARAMETER :: Base_Temp = 50
  - REAL, PARAMETER :: Pi = 3.141593, TwoPi = 2.0*Pi
  - CHARACTER(2), PARAMETER :: Units = “cm”
  - CHARACTER, PARAMETER :: Units = “cm”
  - How is this used? Examples

Operations and Functions

- Operators: +, -, *, /, **
- Numeric operations
  - 3.0 + 4.0 = 7.0, 9.0/4.0 = 2.25
  - 3 + 4 = 7, 9 / 4 = 2
- Mixed mode operations
  - When an integer quantity is combined with a real one, the integer quantity is converted to its real equivalent, and the result is of that type real.

Mixed Modes of Operations

- 1.0/5 → 1.0/5.0 → 0.20
- 32.0 + 9/5 → 32.0 + 1 → 32.0 + 1.0 → 33.0
- 32 + 9.0/5 → 32 + 9.0/5.0 → 32 + 1.8 → 32.0+1.8 → 33.8
- Last two are algebraically equal, but not equal due to difference between integer and real arithemetic.

Priority Rules

1. Exponentiations (right to left)
2. Multiplication and division (left to right)
3. Additions and subtractions (left to right)

- Only expressions in which operands of different types should be used are those in which a real value is raised to an integer power.

  – 2.0 **3 → 2.0 * 2.0 * 2.0 → 8.0
  – (-4.0) **2 → (-4.0) * (-4.0) → 16.0

Examples

- 4+8**2 / 2 =
- Order of operation can be modified by used parenthesis
  - (4+8**2) / 2 =
  - (2+3**2) / (8-2+1) =
  - (2.0+3**2) / (8-2+1) =

- Intrinsic Numeric Functions
  - Table 2-2
  - Appendix A

Character Operations

- Concatenation Operator: //
  - Example: “kilo” // “meter” = “kilometer”
  - Example: Unit = “square_”

    Unit // “kilo” // “meter” = “square kilometer”

- Substring
  - Unit = “kilometer”
  - Unit(5:7) = “met”

Assignment Statement

```
REAL :: XCoordinate, YCoordinate
INTEGER :: Number, Term
XCoordinate = 5.23
YCoordinate = SQRT(25.0)
Number = 17
Term = Number / 3+2
XCoordinate = 2.0 * XCoordinate

CHARACTER(5) :: Truncated, Padded*10
Padded = “Frost”
Truncated = “Temperature”
```

Output

- Output
  - List-directed, formatted
  - PRINT *, output-list
  - WRITE(*,*) output-list
- Class example
- PRINT * and WRITE(*,*) produce blank lines

List-directed Input

- READ *, input-list
- READ(*,*) input-list

- A new line of data is processed each time a read statement is executed.
- If there are fewer entries in a line of input data than there are variables in the input list, successive lines of input are processed until all variables in the list have been obtained.
- If there are more entries in a line of input than there are variables in the input list, the first data values are used.
- The entries in each line of input data must be constants and of the same type as the variables to which they are assigned.
  - Auto-type conversion does take place.

Program Composition and Format

- Heading
  - Program heading
  - Opening documentation
- Specification
- Execution
- Subprogram
- END Program statement

Heading

- PROGRAM name
  - Name is a legal FORTRAN identifier
  - Marks the beginning of the program and gives it a name.
- Opening documentation
  - Explains the purpose of the program
  - Contains variable list
  - Provides other information about the program
- Comments/documentation in FORTRAN 90 begin with a exclamation mark (!)
  - ! This is a line of comment. It will be ignored by the compiler.
  - Use comments to clarify the purpose and structure of key parts of the program.

Specification

- First line: IMPLICIT NONE
- Contains variable declarations (type statements)
  - Should specify each variable and constant used in the program.
- INTEGER :: Loop
- INTEGER :: Loop ! Loop variable

Execution/Subprogram/END Program

- Execution
  - Contains statements that specify actions to be performed during execution of the program.
  - Be sure to use correct syntax
    - Syntax: grammatical rules of a language.
- Subprogram
  - Contains internal Subprograms
  - More on this later
- END PROGRAM statement
  - END PROGRAM name
  - Indicates the end of the program
  - Halts execution of the program

Putting it all together

- Write program
  - Done using your favorite editor.
  - Comment, comment, comment!
- Compile program
  - GNU FORTRAN compiler (gfortran).
  - Fix compile time errors.
- Execute Program
  - Fix run time errors.
  - Evaluate results, fix errors.

Practice

- Read the ‘Program Style and Design’ section on page 40 and the ‘Potential Problems’ section on page 41.

- Write a FORTRAN 90 program that will print ‘Hello World’ to standard output (screen).

- To write a program a that is a little more interesting, build a simple temperature conversion program. Ask the user to input a temperature in a specified unit. Convert the temperature and output all of the data for the user to read on the screen.