Variables and Data Types

Carlos Cruz
Jules Kouatchou
Bruce Van Aartsen

NASA GSFC Code 606 (ASTG)
Greenbelt, Maryland 20771

October 24, 2018

Variables

- Variables are used to hold values and then write mathematical expressions using them
- A variable can hold one value at a time.
- Variables must be declared at the start of the program.
- Each variable must be named. The variable name is how variables are referred to by the program.
- A variable name must start with a letter, followed by letters, numbers, or an underscore.
- A variable name may not be longer than 32 characters.
- Capital letters are treated the same way as lower-case letters

Examples of Variable Names

Some valid variable names:

```
x
today
next_month
summation10
```

Some invalid variable names:

```
1today
this_is_a_variable_name_with_way_way_too_many_characters_i
next@month
next month
today!
```

SSAI

Variable Declaration

```
1     <type> :: <list of variable names>
```

The type must be one of the predefined data types.

Data Types

Fortran has five intrinsic data types:

- Integer
- Real
- Complex
- Logical
- Character

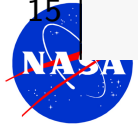
You can derive your own data types as well.

Integer Data Type

- Can hold only integer values (whole numbers)
- You can specify the number of bytes using the *kind* specifier
- For integer division, no rounding will occur as the fractional part is truncated

Examples of Integer Data Type

```
1  program testingInteger
2  implicit none
3
4     integer(kind = 2) :: shortval !two byte integer
5     integer(kind = 4) :: longval !four byte integer
6     integer(kind = 8) :: verylongval ! eight byte integer
7     integer(kind = 16) :: veryverylongval ! sixteen byte int
8     integer :: defval ! default integer
9
10    print *, huge(shortval)
11    print *, huge(longval)
12    print *, huge(verylongval)
13    print *, huge(veryverylongval)
14    print *, huge(defval)
15 end program testingInteger
```



Real Data Type

- Also referred to as floating-point numbers, include both rational numbers and irrational numbers
- There are two different real types, the default *real* type and *double precision* type
- You can specify the precision of real using the *kind* specifier

Examples of Real Data Type

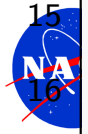
```
1  program testingReal
2  implicit none
3     real(kind=4):: real4Res ! Define single precision real v
4     real(kind=8):: real8Res ! Define double precision real v
5     integer :: intRes ! Define integer variables
6
7     ! floating point division
8     real4Res = 2.0/3.0
9     real8Res = 2.0/3.0
10    intRes = 2/3
11
12    print *, 'Single precision division:', real4Res
13    print *, 'Double precision division:', real8Res
14    print *, 'Integer division:', intRes
15 end program testingReal
```

Complex Type

- Used to store complex number (number comprising a real number and an imaginary number).
- Not used extensively, but can be useful when needed.

Example of Complex Type

```
1  program testingComplex
2  implicit none
3     real :: x, y
4     complex :: cx1, cx2, cx3
5
6     x = 2.67
7     y = -0.349
8     cx1 = (3.0, 5.0) ! cx1 = 3.0 + 5.0i
9     cx2 = cmplx (1.0/2.0, -7.0) ! cx2 = 0.5  7.0i
10    cx3 = cmplx (x, y) ! cx3 = x + yi
11    print *, 'cx1: ', cx1
12    print *, 'cx2: ', cx2
13    print *, 'cx3: ', cx3
14    print *, 'cx1+cx3: ', cx1+cx3
15    print *, 'cx1-cx2: ', cx1-cx2
16    print *, 'cx1*cx2: ', cx1*cx2
17    print *, 'cx1/cx2: ', cx1/cx2
```



Logical Type

- Stores logical variable
- Takes two possible values: *.true.* or *.false.*.

Example of Logical Type

```
1  program testingLogical
2  implicit none
3  INTEGER :: YEAR
4  LOGICAL :: LEAP_FLAG
5
6  YEAR = 2004
7  LEAP_FLAG = .FALSE.
8  IF (MOD(YEAR,4) .EQ. 0) LEAP_FLAG = .TRUE.
9  IF (MOD(YEAR,100) .EQ. 0) LEAP_FLAG = .FALSE.
10 IF (MOD(YEAR,400) .EQ. 0) LEAP_FLAG = .TRUE.
11
12 PRINT*, 'Is ', YEAR, ' a leap year? ', LEAP_FLAG
13 end program testingLogical
```

Character Type

- Stores a character (symbol like a letter, numerical digit, or punctuation) and a string (sequence or set of characters).
- Characters and strings are typically enclosed in quotes.
- The length of the string can be specified by *len* specifier
- If no length is specified, then *len=1*.

Examples of Character Data Type

```
1  program testingCharacter
2  implicit none
3  integer, parameter :: maxLengthChar = 50
4  character (len = maxLengthChar) :: stationName
5  character (len = maxLengthChar) :: welcomeMsg
6  character (len = maxLengthChar) :: location
7  character :: oneChar
8
9  welcomeMsg = ' Welcome to the Fortran Tutorial '
10 location = 'Hampton, Virginia''
11 stationName = 'Blodgett'
12 oneChar = 'T'
13 print *, 'Message - 1: ', welcomeMsg
14 print *, 'Message - 2: ', TRIM(welcomeMsg)
15 print *, 'Message - 3: ', TRIM(welcomeMsg)//' in '//TRIM(lo
16 print *, 'Name of station: ', stationName
17 print *, oneChar
```