Data Types and Intrinsic Functions

22.901 Introduction to Computer Programming for Nuclear Engineers

January 19, 2012

Massachusetts Institute of Technology

Outline

- Intro to programming

Ensuring Explicit Variable Declaration

- Lecture 1 highlighted that Fortran has implicit type designation depending on the first letter
- In reality, we do not want to be restricted to this so we will *declare* all varibles
- When declaring variables we give the type, attributes, comments and other options
- If we forget to declare a variable, Fortran will use the implicit naming scheme!
  - What is the danger in this?
- To ensure that no variable is declared implicitly we use `implicit none`
- This statement should be listed a the top of every routine!

Massachusetts Institute of Technology

DataTypes

- **integer** for exact whole numbers
- **real** for approximate, fractional numbers
- **complex** for complex, fraction numbers
- **logical** for boolean values `.true.` and `.false.`
- **character** for strings of characters
- Note: strings are constructured from an *array* of characters

Massachusetts Institute of Technology

When to use Integer vs. Real

Integers::

- Always think to yourself is this variable always a whole number?
- Loop counts or loop limits
- An index into an array
- For error codes

Reals::

- Are held as floating-point values
- Can be defined with decimals and exponents

Massachusetts Institute of Technology

How to declare a variable

```
integer ::  i !  loop iteration counter for x
real ::  flux !  scalar for total neutron flux
logical ::  finished = .false.  !  code completed?
character(len=8) ::  astr !  a string
```

- All of the different type of declarations are above
- You can set values right in the declarations
- The '::' symbol separates the type info from the variable
- Single vs. Double Precision
  - Default is 4 bytes (single precision) real $\equiv$ real(4)
  - To get double precision use real(8), 8 bytes

Massachusetts Institute of Technology

Type Conversions

In any expression it is important to have the same variable type!

- `real(arg)` – converts `arg` to a real(4)
- `int(arg)` – converts `arg` to a integer(4)
- `dble(arg)` – converts `arg` to a real(8).
- `ceiling(x)` – smallest integer greater or equal to `x`
- `floor(x)` – largest integer less or equal to `x`
- `nint(x)` – nearest integer to `x`

Massachusetts Institute of Technology

Homework 1 w/ Declarations I

```
program assign1

implicit none

  integer :: n ! integer to read in
  integer :: i ! the output variable

  ! ask user for whole number 1-9
  print *,'Enter number 1-9::'
  read *,n

  ! multiply by 9
  i = 9*n

  ! add digits together
  i = i/10 + mod(i,10)

  ! subtract by 5
```

Massachusetts Institute of Technology

Homework 1 w/ Declarations II

```
  i = i - 5

  ! print out result
  print *,'The result is:',i

  ! terminate the program
  stop

end program assign1
```

Fortran Intrinsic Functions

Fortran intrinsic functions are built-in routines that you can call

- `y = sqrt(x)`
- `pi = 4.0*atan(1.0)`
- `z = exp(3.0*y)`
- `u = log(x)` natural log use `log10` for base 10
- `sin(x),cos(x),tan(x)` ... angles are in rads
- `max(x,y), min(x,y)`

There are so many, please search internet if you are looking for something specific!

Massachusetts Institute of Technology

Character Type

- Used when strings of characters are required
- Fortran's basic type is a fixed-length string
- Character constants are quoted strings
  - `print *, 'This is a title'`
  - `print *, ''And so is this''`
- The characters between quotes are the value

Character Variables

```
character ::  answer, marital_status
character(len=10) ::  name,dept,faculty
character(len=32) ::  address
```

`answer` and `marital_status` are each of length **1**.
They hold precisely one character each such are 'Y' or 'n'.

`name`, `dept` and `faculty` are of length **10**.
`address` is of length **32**.

If you do not use the full 'len' the rest is whitespace!

Character Concatenation

Strings may be joined using the // operator

```
character(len=6) ::  identity,a,b,z
identity = 'TH' // 'OMAS'
a = 'TH'
b = 'OMAS'
z = a // b
print *,identity
print *,z
```

What is the result?

// does not remove trailing spaces!

Massachusetts Institute of Technology

Character Intrinsics

```
len(c) -- the storage length of c
trim(c) -- remove trailing blanks
adjustl(c) -- remove leading blanks
index(str,sub) -- position of a substring in string
repeat(str,num) -- copy a str num times and join
```

Again there are many intrinsics to search internet!

```
z = trim(a)//trim(b)
print *,z
```

This will give us THOMAS!

Massachusetts Institute of Technology

Parameters or Constants

- `parameter` is an optional attribute for a variable
- parameters cannot be changed during the execution of the code
- the actual values of the parameters are specific and substituted in a compile time
- this can reduce errors for variables that should not change

```
real(8), parameter ::  pi = 3.14159_8 (or 3.14159D0)
integer, parameter ::  maxlen = 100
```

Massachusetts Institute of Technology

Calculating the Hypotenuse I

```
program pythag

implicit none

  real                  :: hyp      ! the measure of the hyp.
  real                  :: leg1     ! the measure of first leg
  real                  :: leg2     ! the measure of second leg
  character(len=10)     :: unitstr  ! the unit

  ! read in both legs from user
  print *, 'Enter both legs of right triangle::'
  read *, leg1, leg2

  ! have the user enter units
  print *, 'Enter units of legs'
  read *, unitstr

  ! perform pythagorean theorem
```

Calculating the Hypotenuse II

```
    hyp = sqrt(leg1**2 + leg2**2)

    ! print result
    print *,'The hypotenuse is:',hyp,' '//trim(unitstr)

end program pythag
```

End of Class/External Assignment

Law of Cosines

- Declare all variables
- Get user input for 2 sides and an angle between (read in degrees)
- Also from user get name of the third side (The length (sidename) is)
- Do out the math (be careful!)
- Print results to user