FORTRAN – BASICS

Victor Eijkhout, Harika Gurram, Je'aime Powell, Charley Dey

Fall 2018



History

- Fortran stands for Formula Translation
- Designed with the scientist in mind
- First high-level computer language, circa 1956
- https://en.wikipedia.org/wiki/Timeline_of_program ming_languages



Usage

- Compiled
 - Intel compiler (preferred)
 - ifort sourcefilename.F90 -o outputfilename
 - GNU compiler
 - gfortran sourcefilename. F90 -o outputfilename



Program structure

```
Program foo

< declarations >

< statements >

End Program foo
```

Jumping In - Code this now.

Hello World

```
program hello
```

implicit none

print *, 'Hello World'

end program hello



Jumping In

Hello World... what's different?

program hello
implicit none
print *, 'Hello World'
end program hello

What's different from C/C++?

- no int main()
- no curly braces
- no semi-colons (semi-colons can be used to put multiple statements on one line
- program begins with program programname
- implicit none statement
- separate variable declaration section and execution section
- program end with end program programname
- ..
- ..
- more to come...



Statements

• One line, one statement

$$x = 1$$
$$y = 2$$

• semicolon to separate multiple statements per line

$$x = 1; y = 2$$

Continuation of a line

```
x = very &
  long &
  expression
```



Comments

• Ignore to end of line

```
x = 1! set x to one
```

• comment after continuation

```
x = f(a) & ! term1 + g(b) ! term2
```



Jumping In

Hello World

program hello

implicit none

print *, 'Hello World'

end program hello

Start with:

program <program name>

Declaration section

Turn-off implicit declarations:

implicit none

Execution section

Print to screen:

print *, 'text'
*: Automatic formatting

End with:

end program program name>



Jumping In

Hello World with comments and continuation

```
program hello
                                                  Comments start with!
                                                        ! This is a comment
! This is a comment
! Comments start with an
    exclamation mark (!)
                                                  Comments start with!
! This program prints
                                                        print * ! comment starts after !
  "Hello World" on the screen
! Turn off implicit declarations
                                                  Continue a line with &
implicit none
                                                        print *, &
print *, 'Hello World' ! print
                                                           'Hello World'
! with a continuation line
! Last character is a &
print *, &
 'Hello World'
end program hello
```



Exercise 1 & 2

Hello World

Take the 'hello world' program you wrote earlier, and duplicate the hello-line. Compile and run.

Does it make a difference whether you have the two hellos on the same line or on different lines?

Experiment with other changes to the layout of your source.

Find at least one change that leads to a compiler error.



Variable declarations

- Variable declarations at the top of the problem
- Variables are implicitly defined. Dangerous, so use:
- declaration

```
type, attributes :: name1, name2, ....
where
```

- type is most commonly integer, real(4), real(8), logical
- attributes can be dimension, allocatable, intent, parameters et cetera.



Implicit typing

Fortran does not need variable declarations: type are determined by name.
This is very dangerous. Use:

implicit none

in every program unit.



Single precision constants

```
real(8) :: x
x = 3.14
y = 6.022e-23
```

Double precision constants

Use a compiler flag such as -r8 to force all reals to be 8-byte.

Write 3.14d0

```
x = real(3.14, kind=8)
```



Floating point types

Indicate number of bytes:

```
integer(2) :: i2
integer(4) :: i4
integer(8) :: i8

real(4) :: r4
real(8) :: r8
real(16) :: r16

complex(8) :: c8
complex(16) :: c16
complex*32 :: c32
```



Numerical precision

Number of bytes determines numerical precision:

- Computations in 4-byte have relative error $\approx 10^{-6}$
- Computations in 8-byte have relative error $\approx 10^{-15}$

Also different exponent range: max 10^{50} and 10^{300} respectively.



Complex

Complex constants are written as a pair of reals in parentheses. There are some basic operations.

Code:

Output from running complex in code directory basicf:



Arithmetic expressions

- Pretty much as in C++
- Exception: r**2 for power.
- Modulus is a function: MOD(7,3).

+	addition
-	subtraction
*	multiplication
1	division
**	exponent



Boolean expressions

- Long form .and. .not. .or. .lt. .eq. .ge. .true. .false.
- Short form: < <= == /= > >=



I/O routines

• Input:

READ *,n

• Output:

PRINT *,n

There is also WRITE.

Other syntax for read/write with files and formats.



Variables and Assignments

```
program variables
                                             Declaration section
                                             Integer variables
implicit none ! Declaration
                                                  integer :: var1, var2
integer :: year, day ! Section
                                             Real variables
real :: age
                                                  real :: var3, var4
year = 2010
               ! Execution
day = 9
                  ! Section
                                             Execution section
age = 27.35
                                             Assignments
print *, 'year', year
                                                  variable = value
print * ! Print a blank line
                                             Real assignment with a decimal
print *, 'This is day', day
print *, 'She is', age, 'years old'
                                                  var3 = 17.5
                                                  var4 = 18.
end program variables
                                             Integer assignments
                                                  var1 = 17
```



Constants and Expressions

```
Declaration section
program variables
                                                    Integer variables
implicit none
                                                          integer :: var1, var2
real
                  :: age, years left
                                                    Real constant
real, parameter :: ret age = \overline{62}.
                                                          real, parameter :: &
                                                             const = <value>
! Assign the age
           = 27.35
                                                    Execution section
! Calculate the years to retirement
                                                    Assignments
years_left = ret_age - age
                                                          variable = <variable>
print *, 'Years to retirement:', &
                                                    Expression
          years left
                                                          variable = <expression>
                                                    Examples
end program variables
                                                          i = 5
                                                          x = 2.5 * v
                                                          a = b + c
```



Rules: Variables, Declarations, Assignments

- Names in Fortran are between 1 and 31 characters in length
- Names are case-insensitive
 - Var, vAr, VAR, and var are equivalent names
- First character in a name must be an alphabet character; names must not start with a number
- Names must not contain non-alphanumeric characters (but the underscore can be used)
- NOTE: If implicit none is not specified in a program
 - variables with names that begin with the letters i-n are integer by default
 - variables with names that begin with a-h or o-z are of type real by default



Assignments and Expressions Example

```
program assign
implicit none
real
        :: x, y
integer :: i, j
x = 3.4
x = 2.*x
                       ! Evaluate Right-Hand-Side first
                       ! then assign result to Left-Hand-Side
y = 4.*x*x + 2.5*x - 3.4 ! 3.4, 4. and 3.4 are unnamed constants of type real
i = 4
                       ! 4 and 2 are unnamed constants of type integer
i = 2*i
i = 2*i*i + 4*i - 2
y = i * x
                      ! i is converted into a real before the calculation
y = real(i) * x ! Explicit type conversion with the function real()
end program assign
```



Rules: Variables, Declarations, Assignments

Type [Optional attributes] :: Variables

```
integer[ kind selector ]
real[ kind selector ]
complex [ length selector ]
logical[ length selector ]
character[ length selector ]
```

- kind specifies how many *bytes* the variable will require
 - usage: kind=integer value
- length specifies how *long* the variable is
 - usage: len=integer value

- Other optional attributes :
 - parameter, allocatable, dimension, intent, optional, save, pointer, target



Data Types, Assignments and Expressions Example



Data Types, Assignments and Expressions Example



Exercise 3

Variables, Declarations, Assignments

Write a program that has several variables of different types Assign values either in an initialization or in an assignment. Print out the values.



Reading input from the keyboard

Execution section

Read from Keyboard

read *, <variable>

Examples

read *, input read *, age read *, age1, age2

Exercise 4

Variables, Declarations, Assignments

Take your program from Exercise 3
Assign the values using the keyboard
Print out the values.



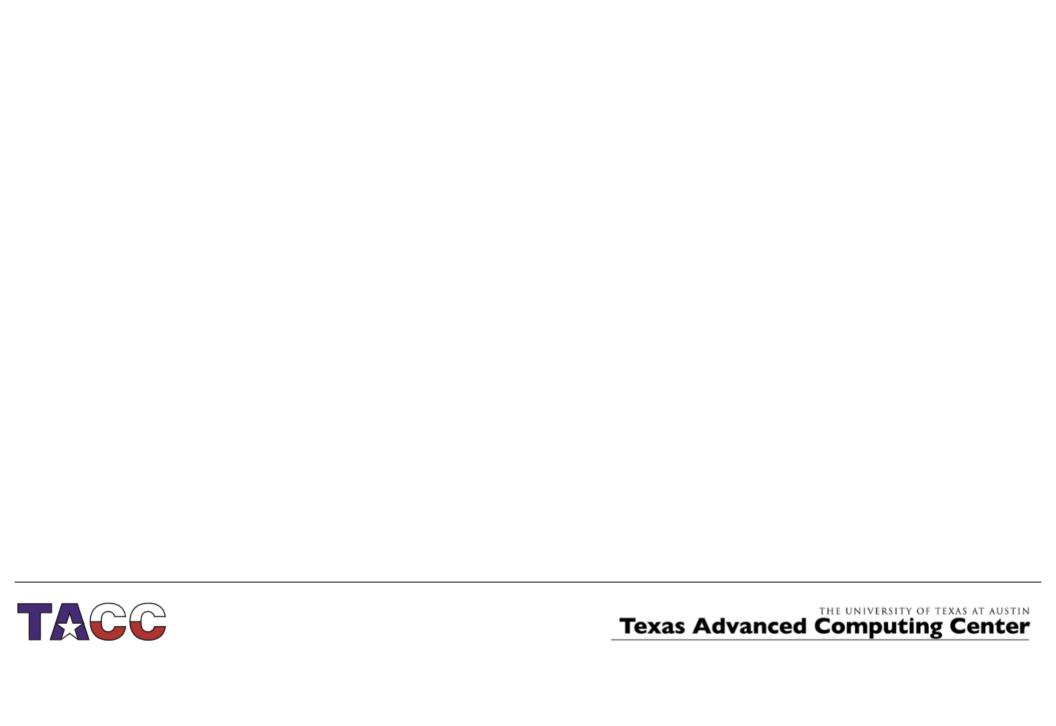
Exercise 5

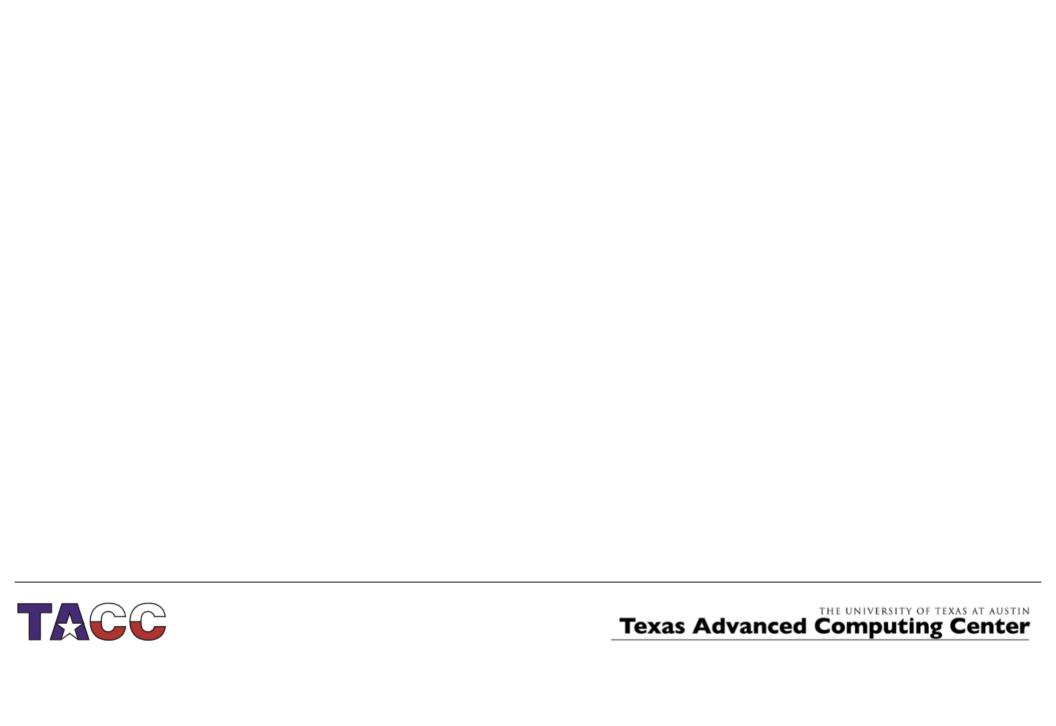
Variables, Declarations, Assignments

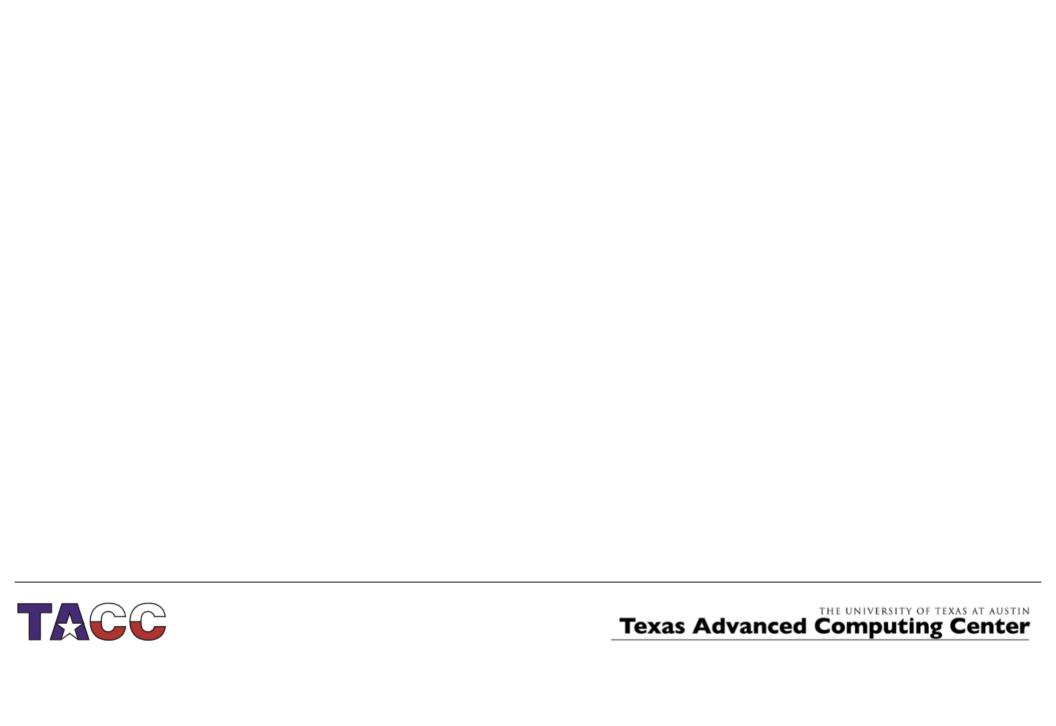
Write a program that accepts three numbers, (a, b, and c) from the keyboard and your name (name)

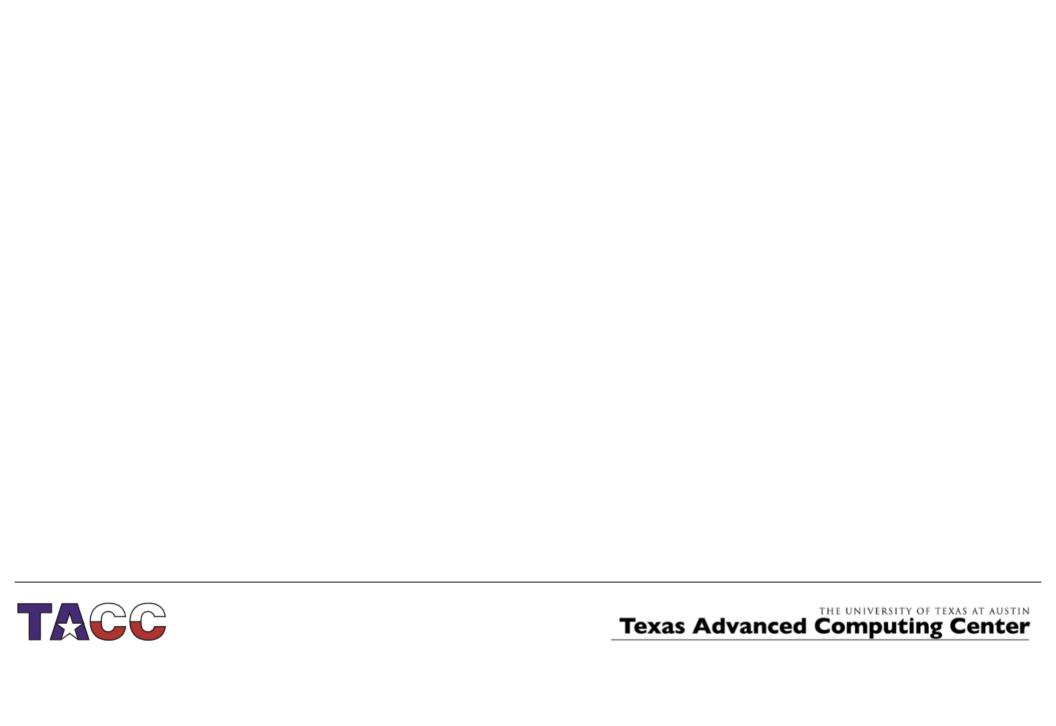
- The program will the say hello to you i.e. "Hello, Jim"
- It will then calculate the volume of a sphere with a being the radius.
 - V = (4/3) * pi * a^3 (NOTE: the 2 *'s are used for exponent, i.e. a^3 would be a**3.0)
- Calculate the volume of a cube
 - 'a' being the length,
 - 'b' being the height,
 - 'c' being the width.
- BONUS:
 - create a real data type, d.
 - set d = (a * b * c)/7.
 - convert d to an integer.

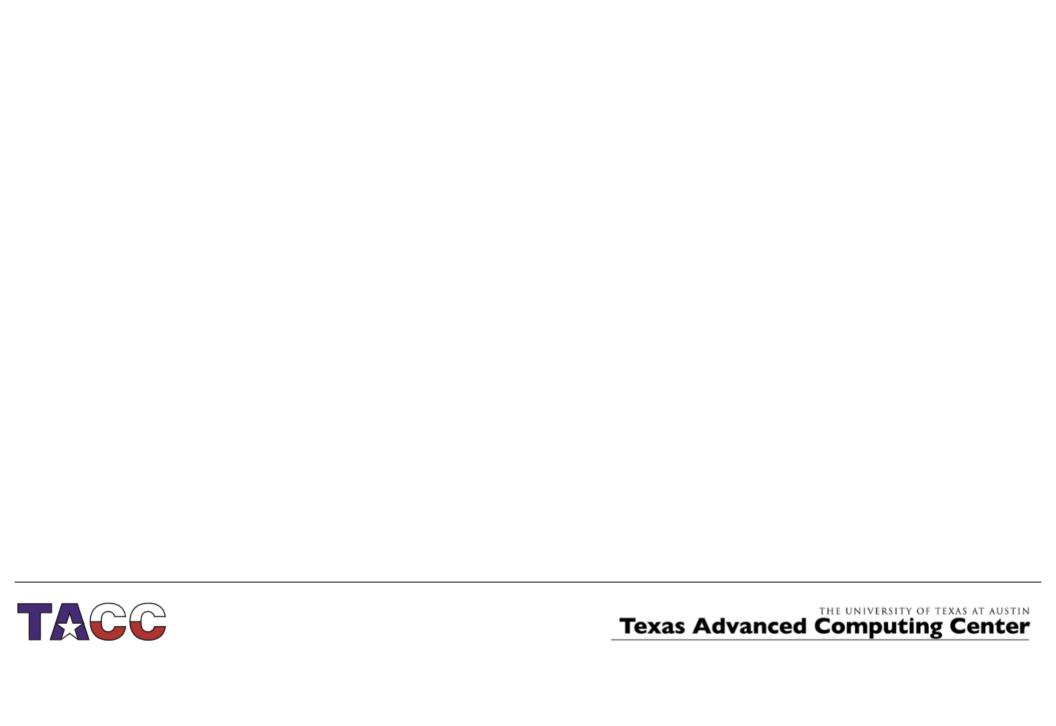


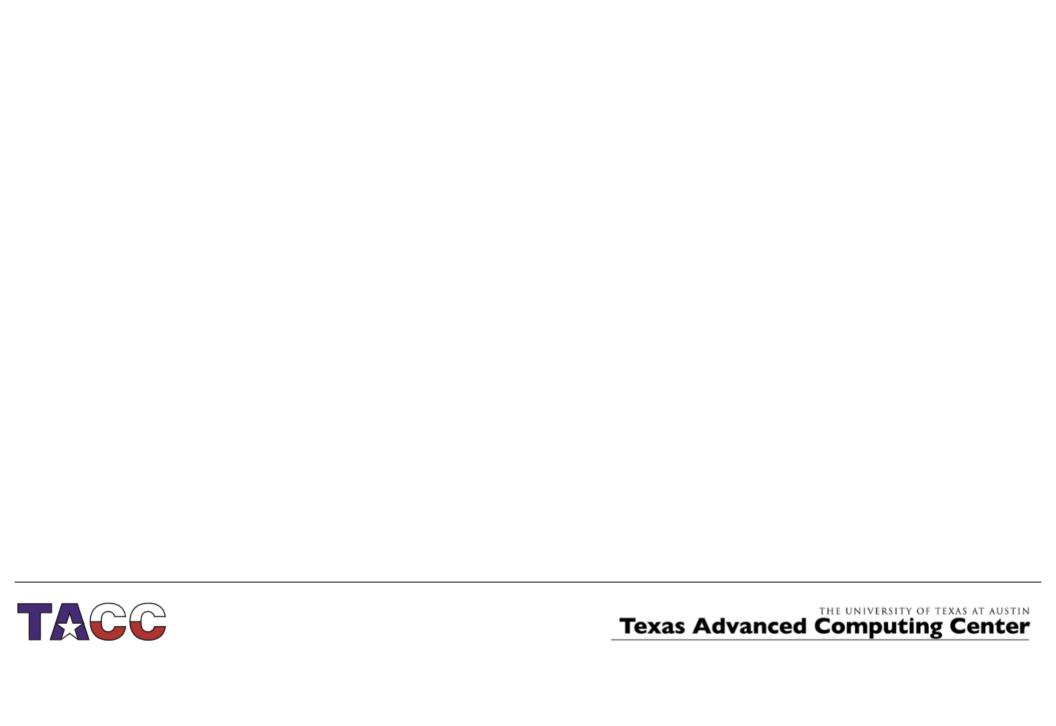












Exercise 1 & 2

Hello World

Take the 'hello world' program you wrote earlier, and duplicate the hello-line. Compile and run.

Does it make a difference whether you have the two hellos on the same line or on different lines?

Experiment with other changes to the layout of your source.

Find at least one change that leads to a compiler error.

Experiment with the print statement.

Replace the string by a number or a mathematical expression.

Can you guess how to print more than one thing, for instance the string One third is and the result of 1/3, with the same print statement?

