Version 2014-1
as of 1 February 2014

Introduction to Scientific Programming

Fortran Language

Part 1

THE UNIVERSITY OF TEXAS AT AUSTIN
Texas Advanced Computing Center

Prefix

Fortran = ***Formula Translation***

- Designed for scientific computing
- First high-level computer language (1956)

  - New Fortran: Fortran90 and later versions: 95, 2003, 2008
  - Use file extension `.f90`

- A compiled language (in contrast to interpreted languages)
  - compile: `ifort source.f90` → `a.out`
  - execute: `./a.out`

Organization

1. Present example code
   - Code is written in some kind of English
   - At the beginning, just by reading you will get most of the ideas
2. More details and some fine print
3. Learning by doing (homework, etc.)
4. By looking at examples in books or the web
5. By asking us

Note: The beginning will be a bit slow for some of you, but we have to level the playing field for those who have never written a single line of code!

Fortran Part 1

- Example code
  - Program, Implicit none, End program
  - Print to screen
  - Comments, Continuation lines
  - Variables, Constants, Assignments
  - Expressions
  - Read from keyboard
  - Assignments
- Full story
  - Variables of type: Integer, Real, Logical, Character, Complex
  - Expressions and Assignments

First Program: Hello World

Program, Implicit None, Print

```
program hello

implicit none

print *, 'Hello World'

end program hello
```

Homework:
All programs that you turn in need to use **`implicit none`**

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>
```

Comments, Continuation Lines

```
program hello

! This is a comment
! Comments start with an
!     exclamation mark (!)
! This program prints
!    "Hello World" on the screen

! Turn off implicit declarations
implicit none

print *, 'Hello World' ! print

! with a continuation line
! Last character is a &
print *, &
  'Hello World'

end program hello
```

Comments start with !

```
! This is a comment
```

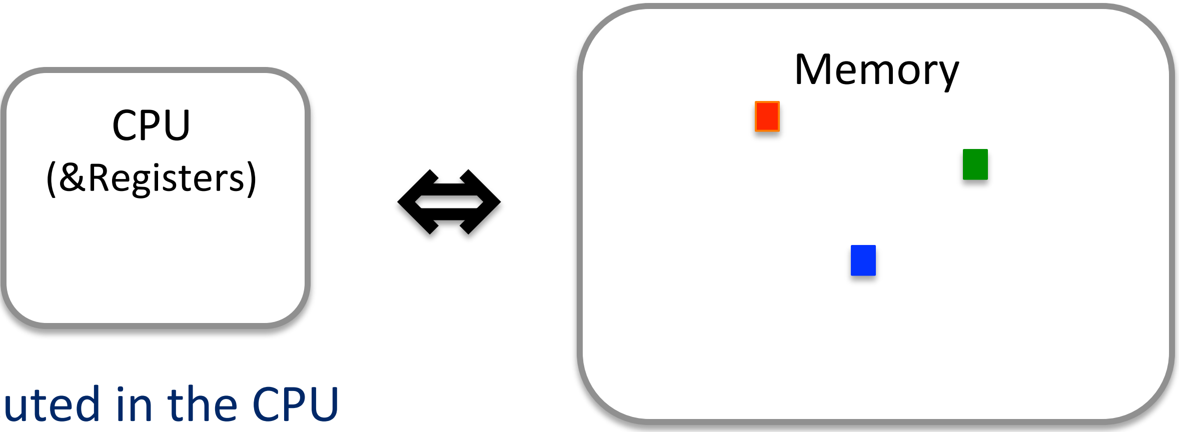
Comments start with !

```
print * ! comment starts
        after !
```

Continue a line with &

```
print *, &
   'Hello World'
```

Computer Architecture: Memory



- Code is executed in the CPU
- Data is stored in Memory
- High-Level Language (Fortran, C++/C, etc.)
  - Programmer not concerned where data is stored
  - Variables are used

Variables are stored somewhere: a, b, c

From Fortran 95/2003 for Scientists and Engineers:

*"A **Fortran variable** is a data object that can change value during the execution of a program. When a Fortran compiler encounters a variable, it reserves a known location in memory for the variable, and then references that memory location whenever the variable is used in the program."*

Basic Concepts

- a, b and c are "Variables" as explained on the previous slide

```
a = 5.5    ! Assignment: Copy the value 5.5 to the variable a
           !   a <== 5.5
b = a + 1. ! Not a mathematical equation, but again an assignment
           !   Evaluate the Right-Hand-Side (RHS) and copy the result to the
           !   variable on the Left-Hand_side (LHS)
           !   b <== (a + 1.)
           ! In this example: copy 6.5 to b
a = a + 2. ! The same variable can appear on the RHS and the LHS
           !   again: evaluate RHS first, and copy result to LHS
           !   a <== (a + 2.)
           !   In this example: copy 7.5 to a
c = a + b  ! Another example: c <== (a + b)
```

Variables and Assignments

```
program variables

implicit none           ! Declaration
integer :: year, day    ! Section
real    :: age

year = 2010             ! Execution
day  = 9                ! Section
age  = 27.35

print *, 'year', year
print *   ! Print a blank line
print *, 'This is day', day
print *, 'She is', age, 'years old'

end program variables
```

Output:

```
 year         2010

 This is day           9
 She is    27.35000    years old
```

Declaration section

Integer variables

```
integer :: var1, var2
```

Real variables

```
real    :: var3, var4
```

Execution section

Assignments

```
variable = value
```

Real assignment with a period

```
var3 = 17.5
var4 = 18.
```

Integer assignments

```
var1 = 17
```

Constants and Expressions

```
program variables

implicit none
real              :: age, years_left
real, parameter :: ret_age = 62.

! Assign the age
age          = 27.35
! Calculate the years to retirement
years_left = ret_age - age

print *, 'Years to retirement:', &
         years_left

end program variables
```

```
Output:
 Years to retirement:   34.65000
```

Declaration section

Integer variables

```
integer :: var1, var2
```

Real constant

```
real, parameter :: &
             const = <value>
```

Execution section

Assignments

```
variable = <variable>
```

Expression

```
variable = <expression>
```

Examples

```
i = 5
x = 2.5 * y
a = b + c
```

Read from Keyboard

```
program read

implicit none
real             :: input
real, parameter :: ret_age = 62.

! Read from Keyboard
print *, 'Enter your age:'
read  *, input
print *, 'You have entered', input

! Calculate the years to retirement
years_left = ret_age - input ! simple
                             ! expression

print *, 'Years left', years_left

end program read
```

Output:

```
 Enter your age:
 33.7
 You have entered   33.70000
 Years left   28.30000
```

Execution section

Read from Keyboard

```
        read *, <variable>
```

Examples

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

Assignments

```
program assign

implicit none
real                      :: x, y
integer                   :: i, j

x = 3.4                   ! Evaluate Right-Hand-Side first
x = 2.*x                  ! 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
j = 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
```

Character Set & Tokens

- Fortran character set consists of the 26 alphabets, 10 numeric characters and special characters
- e.g., _,+,-,/, *,&,!,.,(),[]

- A token is a combination of alphanumeric characters:

1. keyword (e.g., **if**, **do**, **forall**)
2. label (e.g., **10**) *(we will not use labels, though)*
3. constant (e.g., **3.14**, **-1**, **2.71828**, **"hello"**)
4. name (e.g., **a**, **b**, **var**)
5. build-in function names (e.g., **int**, **real**, **sin**, **dotp**)

Observe the Use of Blank Characters!

- Blanks are allowed to separate tokens, keywords

  `do  i=1,n` requires the space between **`do`** and **`i`**

- Blanks are optional between some keywords, e.g.,

  `end do`, **`end if`**, **`else if`**,

  `enddo`, **`endif`**, **`elseif`**

- Do not use a blank character within a name

Rules on Names

- 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)

We gratefully acknowledge the sponsorship of Chevron Corporation, whose generous support of TACC has made possible this Scientific Computing Curriculum and other student-focused initiatives.

License

When attributing this work, please use the following text: "*Introduction to Scientific Programming* course materials by The Texas Advanced Computing Center, 2014. Available under a Creative Commons Attribution Non-Commercial 3.0 Unported License"