Statements and expressions in Fortran

Victor Eijkhout, Susan Lindsey

COE 322 Fall 2020

Basics

Program structure

Structure of your program file:

```
Program foo
  < declarations >
  < statements >
End Program foo
```

Things to note

- No includes before the program
- Program has a name (emacs tip: type `end<TAB>`)
- There is an `End`, rather than curly braces.
- Declarations first, not interspersed.

Compilation

Filename has extension `.F90` of `.F`.
(this is modern 'free format'; the old 'fixed format' uses extensions `.f90` or `.f`)

Compiler: `ifort` (Intel)
or `gfortran` (Gnu compiler collection)

Statements

- One line, one statement

```
x = 1
y = 2
```

  (historical reasons)

- 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
```

- No multi-line comments.

Variable declarations

- Variable declarations at the top of the program unit, before any executable statements.
- declaration

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

  where
  - *type* is most commonly `integer`, `real(4)`, `real(8)`, `logical`
  - *attributes* can be `dimension`, `allocatable`, `intent`, `parameter` et cetera.
- Keywords and variables are *case-insensitive*

Parameter

Sometimes an identifier corresponds to a constant:
use the `parameter` attribute

```
real,parameter :: pi = 3.141592
```

This can not be changed like an ordinary variable.

Data types

- Numeric: `Integer`, `Real`, `Complex`.
- Logical: `Logical`.
- Character: `Character`. Strings are realized as arrays of characters.

Strings

```
character*20 :: prompt
prompt = "up to 20 characters"
```

Implicit typing

Fortran does not need variable declarations (like python):
variables have a type that is determined by name.
This is **very dangerous**. Use `implicit none` in every program unit.

```
Program myprogram
  implicit none
  integer :: i
  real :: x
  ! more stuff
End Program myprogram
```

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
```

Complex

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

Code:

```
Complex :: &
     fourtyfivedegrees = (1.,1.), &
     other
print *,fourtyfivedegrees
other = 2*fourtyfivedegrees
print *,other
```

Output
[basicf] complex:

```
(1.00000000,1.00000000)
(2.00000000,2.00000000)
```

Arithmetic expressions

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

Boolean expressions

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

Statements

Simple I/O

- Input:

  ```
  READ *,n
  ```

- Output:

  ```
  PRINT *,n
  ```

There is also `Write`.

The 'star' indicates that default formatting is used.
Other syntax for read/write with files and formats.

Exercise 1

Write a program that :

- displays the message `Type a number`,
- accepts an integer number from you (use `Read`),
- makes another variable that is three times that integer plus one,
- and then prints out the second variable.

Optional exercise 2

Write two programs, one that reads a temperature in Centigrade and converts to Fahrenheit, and one that does the opposite conversion.

$$C = (F - 32) \cdot 5/9, \qquad F = 9/5\, C + 32$$

Check your program for the freezing and boiling point of water. (Do you know the temperature where Celsius and Fahrenheit are the same?)

Can you use Unix pipes to make one accept the output of the other?