Statements and expressions in Fortran

Victor Eijkhout and Charlie Dey

spring 2017

Basics

Program structure

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

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
```

Variable declarations

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

  `implicit none`
- declaration

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

  where
  - *type* is most commonly `integer`, `real(4)`, `real(8)`, `logical`. See below; section **??**.
  - *attributes* can be `dimension`, `allocatable`, `intent`, `parameters` et cetera.

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
```

Arithmetic expressions

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

Boolean expressions

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

Statements

I/O routines

- Input:

  ```
  READ *,n
  ```

- Output:

  ```
  PRINT *,n
  ```

There is also `WRITE`.

Other syntax for read/write with files and formats.