Version 2014-1
as of 1 February 2014

Introduction to Scientific Programming

Fortran Expressions and Assignments

Expressions and Assignments

- Scalar numeric expressions
- Type conversion
- Character expressions

Scalar Numeric Expressions

- Expressions involve operators and operands
- Operator action can be binary, acts on two operands;
  or unary, acts on a single operand
- Operands can be of intrinsic, derived or array types

```
(unary operator) operand
```

```
operand (binary operator) operand
```

Operators	
arithmetic	`+ - * /**`
relational	`> >= < <= == /=`
logical	`.not. .or. .and.`

Scalar Numeric Expressions

- Operators follow associativity and precedence rules

  `**, */, +-` in order of precedence

- Expressions are evaluated from left to right, except for the exponentiation operator:

  `x**y**z` is evaluated as `x**(y**z)`

- When in doubt, add brackets!

- Integer division results are truncated towards zero:

  `14/5 = 2` `2**(-3)=0`

Scalar Numeric Expressions

operator	operand type	operand type	result
+, -, *, /, **	I	I	I
+, -, *, /, **	R	R	R
+, -, *, /, **	C	C	C
+, -, *, /, **	I	R	R
+, -, *, /, **	I	C	C
+, -, *, /, **	R	C	C

operator	operand type	operand type	type conversion
<, >, >=, <=, /=	I	I	none
<, >, >=, <=, /=	I	C	I→C
<, >, >=, <=, /=	I	R	I→R
<, >, >=, <=, /=	C	C	none
<, >, >=, <=, /=	R	C	R→C
<, >, >=, <=, /=	R	R	None

Type Conversion

	assigned value
integer	`int(expr, kind(integer_kind))` `int(expr, kind=kind(integer_kind))`
complex	`cmplx(expr ,kind(complex_kind))` `cmplx(expr, kind=kind(comlex_kind))`
real	`real(expr, kind(real_kind))` `real(expr, kind=kind(real_kind))`

```
integer(4)  :: i4
integer(8)  :: i8
real(4)     :: r4
real(8)     :: r8

i4 = integer(r4)
r4 = real(i4)

r8 = real(r4, kind=8)
r8 = real(i8, kind=8) + real(r4, kind=8)
```

logical

- A Logical variable can assume two states: `.true.` and `.false.`
- Common operators are: `==! Equal`

```
\=    ! Not equal
.and. ! logical and
.or.  ! logical or
```

```
logical :: b1, b2, b3

b1 = .true.
b2 = .false.

b3 = b1 ==    b2  ! b3 is .false.
b3 = b1 /=    b2  ! b3 is .true.
b3 = b1 .and. b2  ! b3 is .false.
b3 = b1 .or.  b2  ! b3 is .true.

b3 = (b1 .or. b2) .and. b1  ! b3 is .true
```

character

- Only one operator: //
- // concatenates two character operands

```
character(len=5)  :: a, b
character(len=10) :: c, d

a = 'hello'
b = 'world!'

c = a // b      ! c is "helloworld"
d = a // ' ' // b  ! d is "hello world"
```

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