Tecniche di programmazione in chimica computazionale

Expressions, operations and variables

Emanuele Coccia

Dipartimento di Scienze Chimiche e Farmaceutiche

Constants

- Integer: 2 etc.

Constants

- Integer: 2 etc.
- Floating point: 2.0, 2.d0 etc.

Constants

- Integer: 2 etc.
- Floating point: 2.0, 2.d0 etc.
- String: "Hello, world!"

Constants

- Integer: 2 etc.
- Floating point: 2.0, 2.d0 etc.
- String: "Hello, world!"
- Define a constant using parameter (param.f90)

Variables

- Memory location with three attributes:

Variables

- Memory location with three attributes:
  1. name (case insensitive, ABC equal to aBC)

Variables

- Memory location with three attributes:
  1. name (case insensitive, ABC equal to aBC)
  2. Data type (declared at the beginning of the program)

Variables

- Memory location with three attributes:
  1. name (case insensitive, ABC equal to aBC)
  2. Data type (declared at the beginning of the program)
  3. Value (initializing is good practice): declaration, standard input, assignment

Variables

- Memory location with three attributes:
  1. name (case insensitive, ABC equal to aBC)
  2. Data type (declared at the beginning of the program)
  3. Value (initializing is good practice): declaration, standard input, assignment
- Assignment statement: variable = expression

Variables

- Memory location with three attributes:
  1. name (case insensitive, ABC equal to aBC)
  2. Data type (declared at the beginning of the program)
  3. Value (initializing is good practice): declaration, standard input, assignment
- Assignment statement: variable = expression
- Example assign.f90

Variable declaration

- Explicit declaration

Variable declaration

- Explicit declaration
- Implicit declaration: integer variable if its name begins with (I-N)
- Also possible to customize directives for the implicit declaration

Variable declaration

- Explicit declaration
- Implicit declaration: integer variable if its name begins with (I-N)
- Also possible to customize directives for the implicit declaration
- Better to always start a Fortran code with implicit none

Fortran intrinsic functions

- cos(x): x real in radians

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0
- sqrt(x): x real ≥ 0

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0
- sqrt(x): x real ≥ 0
- acos(x): -1 $\leq$ x $\leq$ 1(radians)

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0
- sqrt(x): x real ≥ 0
- acos(x): -1 $\leq$ x $\leq$ 1(radians)
- mod(x,n): both x and n real or integer

Fortran intrinsic functions

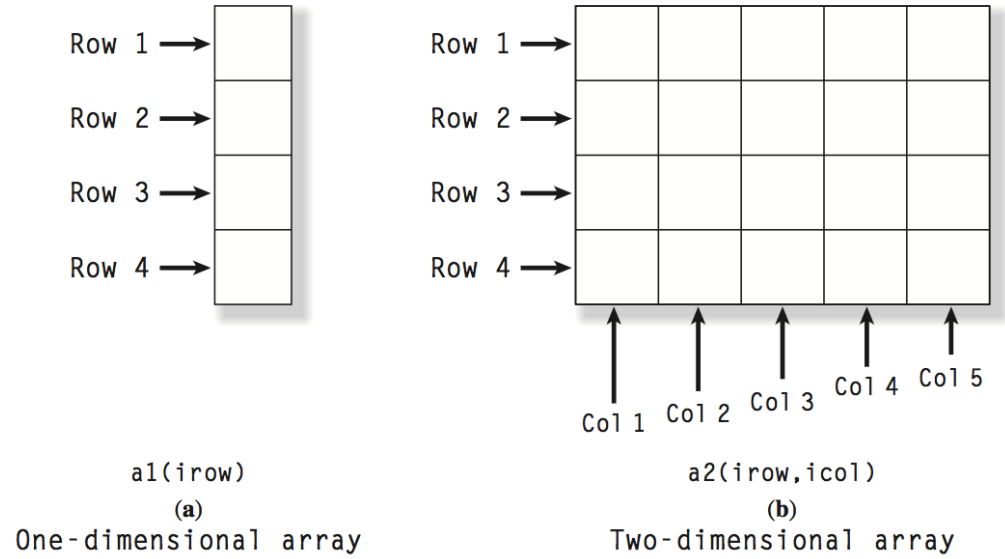
- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0
- sqrt(x): x real ≥ 0
- acos(x): -1 $\leq$ x $\leq$ 1(radians)
- mod(x,n): both x and n real or integer
- ...

Fortran intrinsic functions

- cos(x): x real in radians
- sin(x): x real in radians
- tan(x): x real in radians
- exp(x): x real
- log(x): x real > 0
- log10(x): x real > 0
- sqrt(x): x real ≥ 0
- acos(x): -1 $\leq$ x $\leq$ 1(radians)
- mod(x,n): both x and n real or integer
- ...
- add d (e.g., dcos(x)) for real*8

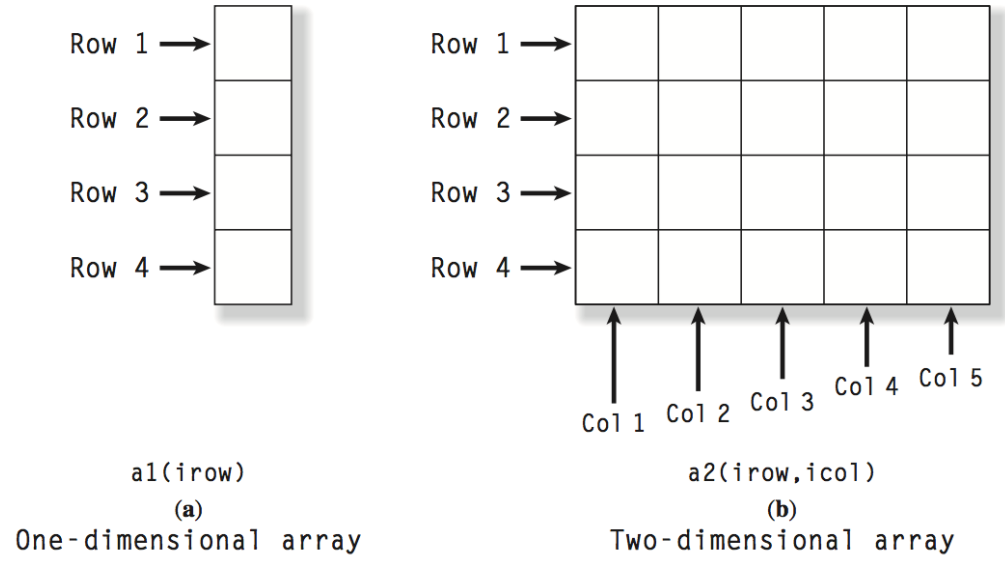
Arrays

- Structured data type: vectors, matrices, more complicated structures



Arrays

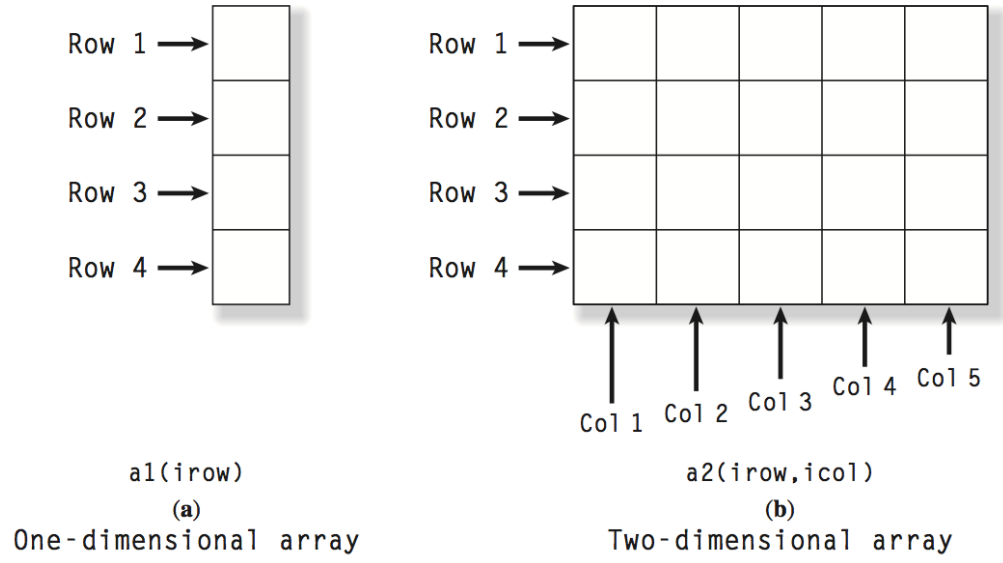
- Structured data type: vectors, matrices, more complicated structures



- Array of any type (integer, single and double precision etc.)

Arrays

- Structured data type: vectors, matrices, more complicated structures



- Array of any type (integer, single and double precision etc.)
- Intrinsic Fortran functions:
  - maxval(x)
  - minval(x)
  - sum(x)
  - dot_product(a,x)
  - matmul(A,B)

Arrays

- Group of ordinary (scalar) variables

Arrays

- Group of ordinary (scalar) variables
- Work on array elements, on a subset of array and/or on the whole array

Arrays

- Group of ordinary (scalar) variables
- Work on array elements, on a subset of array and/or on the whole array
- Initialize arrays as ordinary variables

Arrays

- Group of ordinary (scalar) variables
- Work on array elements, on a subset of array and/or on the whole array
- Initialize arrays as ordinary variables
- Examples array.f90 and matmul.f90

Arrays

- Group of ordinary (scalar) variables
- Work on array elements, on a subset of array and/or on the whole array
- Initialize arrays as ordinary variables
- Examples array.f90 and matmul.f90
- Dynamical arrays: example alloc.f90

Arithmetic expressions

$$n1 + n2$$

- Composed of terms (n1 and n2) and operators (+)
- Computational form to run operations into the computer
- Evaluation: executing the operations specified in a given expression

Arithmetic operators

- Simple arithmetic operators:
  - + addition

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power
- Binary operators:

$$x \quad + \quad 3$$

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power
- Binary operators:

$$x + 3$$
$$2 * x + 3 * y$$

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power
- Binary operators:

$$\begin{aligned} x &+ 3 \\ 2 * x &+ 3 * y \\ (2 * x) &+ (3 * y) \end{aligned}$$

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power
- Binary operators:

$$
\begin{aligned}
x &+ 3 \\
2 * x &+ 3 * y \\
(2 * x) &+ (3 * y) \\
2 * (x &+ 3) * y
\end{aligned}
$$

Arithmetic operators

- Simple arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - ** power
- Binary operators:

$$\begin{aligned} x &+ 3 \\ 2 * x &+ 3 * y \\ (2 * x) &+ (3 * y) \\ 2 * (x &+ 3) * y \end{aligned}$$

- Power > multiplication/division > addition/subtraction

Other operators

- Relational:
  - .eq. (==)
  - .ne. (/=)
  - .lt. (<)
  - .le. (<=)
  - .gt. (>)
  - .ge. (>=)

Other operators

- Relational:
  - .eq. (==)
  - .ne. (/=)
  - .lt. (<)
  - .le. (<=)
  - .gt. (>)
  - .ge. (>=)
- Logical:
  - .not.
  - .and.
  - .or.

Other operators

- Relational:
  - .eq. (==)
  - .ne. (/=)
  - .lt. (<)
  - .le. (<=)
  - .gt. (>)
  - .ge. (>=)
- Logical:
  - .not.
  - .and.
  - .or.
- String: // to concatenate two strings

Other expressions

- Relational expression:

$$n1.\mathrm{eq}.n2$$

Other expressions

- Relational expression:

$$n1.\mathrm{eq}.n2$$

- Logical expression:

$$expr1.\mathrm{or}.expr2$$

Other expressions

- Relational expression:

$$n1.\mathrm{eq}.n2$$

- Logical expression:

$$expr1.\mathrm{or}.expr2$$

- Priority: arithmetic > relational > logical

Type conversion

- Automatic type conversion

Type conversion

- Automatic type conversion
- The expression

$$1 + 2.3$$

Type conversion

- Automatic type conversion
- The expression

$$1 + 2.3$$

in Fortran becomes

$$1.0 + 2.3$$

Type conversion

- Automatic type conversion
- The expression

$$1 + 2.3$$

  in Fortran becomes

$$1.0 + 2.3$$

- Assignment statement for real variable total

$$\text{total} = 0$$

- "0" is converted into "0.0"

Type conversion

- Automatic type conversion
- The expression

$$1 + 2.3$$

  in Fortran becomes

$$1.0 + 2.3$$

- Assignment statement for real variable total

$$\text{total} = 0$$

- "0" is converted into "0.0"

$$n = 1.9$$

- integer n contains "1"

Type conversion

- Automatic type conversion
- The expression

$$1 + 2.3$$

  in Fortran becomes

$$1.0 + 2.3$$

- Assignment statement for real variable total

$$\text{total} = 0$$

- "0" is converted into "0.0"

$$n = 1.9$$

- integer n contains "1"
- Explicit conversion: int(), real(), dble()
- Example conv.f90