PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

Data

Emanuele Coccia

Dipartimento di Scienze Chimiche e Farmaceutiche

Data

- Abstraction of computation modules

Data

- Abstraction of computation modules
- Subroutine/function: solving a specific problem

Data

- Abstraction of computation modules
- Subroutine/function: solving a specific problem
- Subroutine/function and data are distinct entities

Data

- Abstraction of computation modules
- Subroutine/function: solving a specific problem
- Subroutine/function and data are distinct entities
- Procedural programming

Practical example

Given two integers n1 and n2, both read from standard input, we want to sum them into a variable total, then printed on the screen (standard output) (add.f90)

Bit

- Computer memories: billions of individual switches (ON or OFF)

Bit

- Computer memories: billions of individual switches (ON or OFF)
- Binary digit (bit): each switch composing the memory
- ON 1, OFF 1

Bit

- Computer memories: billions of individual switches (ON or OFF)
- Binary digit (bit): each switch composing the memory
- ON 1, OFF 1
- Represent number using groups of bits (i.e., of 0 and 1)

Bit

- Computer memories: billions of individual switches (ON or OFF)
- Binary digit (bit): each switch composing the memory
- ON 1, OFF 1
- Represent number using groups of bits (i.e., of 0 and 1)
- Byte: group of 8 bits

Bit

- Computer memories: billions of individual switches (ON or OFF)
- Binary digit (bit): each switch composing the memory
- ON 1, OFF 1
- Represent number using groups of bits (i.e., of 0 and 1)
- Byte: group of 8 bits
- KB, MB, GB, TB etc.

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)
- real (real*4): approximately represented by a computer using the floating point method

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)
- real (real*4): approximately represented by a computer using the floating point method
- double precision (real*8): same as real but using a double memory

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)
- real (real*4): approximately represented by a computer using the floating point method
- double precision (real*8): same as real but using a double memory
- complex:pair of real (or double precision) numbers

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)
- real (real*4): approximately represented by a computer using the floating point method
- double precision (real*8): same as real but using a double memory
- complex:pair of real (or double precision) numbers
- logical: value .true. or .false.

Data types

Intrinsic data types:

- integer: exactly represented by a computer (from -2^{n-1} to 2^{n-1}-1, n-bit)
- real (real*4): approximately represented by a computer using the floating point method
- double precision (real*8): same as real but using a double memory
- complex:pair of real (or double precision) numbers
- logical: value .true. or .false.
- character*p: string of alphanumeric character, with length p

Data types

- Two properties:
  1. set of values (domain)

Data types

- Two properties:
  1. set of values (domain)
  2. set of operations

Floating point arithmetics

- Two fundamental limitations of integers:
  1. Not possibile to represent fractional numbers

Floating point arithmetics

- Two fundamental limitations of integers:
  1. Not possibile to represent fractional numbers
  2. Not possible to represent very large positive or very small negative numbers

Floating point arithmetics

- Two fundamental limitations of integers:
  1. Not possibile to represent fractional numbers
  2. Not possible to represent very large positive or very small negative numbers
- Manipulating non-integer numbers

Floating point arithmetics

- Two fundamental limitations of integers:
  1. Not possibile to represent fractional numbers
  2. Not possible to represent very large positive or very small negative numbers
- Manipulating non-integer numbers
- Floating point numbers used to approximate real numbers

Floating point arithmetics

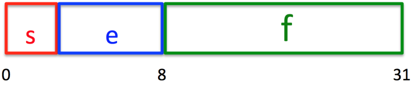
- Two fundamental limitations of integers:
  1. Not possibile to represent fractional numbers
  2. Not possible to represent very large positive or very small negative numbers
- Manipulating non-integer numbers
- Floating point numbers used to approximate real numbers
- Scientific notation in base 2

Floating point arithmetics

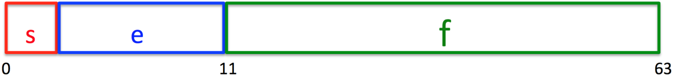
- Single precision (real*4): 32 bits, from $\sim 1.175 * 10^{-38}$ to $\sim 3.403 * 10^{+38}$
- Double precision (real*8): 64 bits, from $\sim 2.225 * 10^{-308}$ to $\sim 1.798 * 10^{+308}$

$$\text{floating point} = (-1)^{s} * f * 2^{e}$$

Single precision



Double precision



Floating point arithmetics

1. Precision:
    - Number of significant digits
    - Depends on the number of bits in f

Floating point arithmetics

1. Precision:
   - Number of significant digits
   - Depends on the number of bits in f
2. Range:
   - Largest and the smallest exponents
   - Number of bits in e
   - Example data.f90

Floating point arithmetics

1. Precision:
   - Number of significant digits
   - Depends on the number of bits in f
2. Range:
   - Largest and the smallest exponents
   - Number of bits in e
   - Example data.f90

- Finite number of significant digits
- Finite range

Floating point arithmetics

1. Precision:
   - Number of significant digits
   - Depends on the number of bits in f
2. Range:
   - Largest and the smallest exponents
   - Number of bits in e
   - Example data.f90

- Finite number of significant digits
- Finite range
- Systematic error in representing real numbers (round-off error)

Base 2

$$101.01 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 5.25$$

Base 2

$$\begin{aligned}
101.01 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 5.25 \\
0.0101 &= 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\
&= 1./4. + 1./16. = 0.3125
\end{aligned}$$

Practical examples

- Rewrite add.f90 with real and double precision numbers