Designing codes

Programming Concepts in Scientific Programming

EPFL, Master class

November 19, 2018

Making scientific software

#1 Scientific question

#2 Problem formulation

Making scientific software

#2 Problem formulation

- Mathematics
- Identify inputs/outputs

#3 Algorithms description

#3 Algorithms description

- Decompose program in sub-parts

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts
- Choose data structures

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts
- Choose data structures
- Identify polymorphic code: class diagram

#4 Implementation

#4 Implementation

- Decide where the code is hosted (for backups and revisions)

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)
  example:
  `https://google.github.io/styleguide/cppguide.html`

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)
  example:
  `https://google.github.io/styleguide/cppguide.html`
- Identify existing software for any sub-part

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)
  example:
  `https://google.github.io/styleguide/cppguide.html`
- Identify existing software for any sub-part
- Decide a source documentation format

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)
  example:
  `https://google.github.io/styleguide/cppguide.html`
- Identify existing software for any sub-part
- Decide a source documentation format
- Program the thing

#4 Implementation

- Decide where the code is hosted (for backups and revisions)
- Decide a coding convention (question of style)
  example:
  `https://google.github.io/styleguide/cppguide.html`
- Identify existing software for any sub-part
- Decide a source documentation format
- Program the thing
- Tests

Making scientific software: an example

#1 Scientific question

Many meteo devices measure constantly the temperature in Switzerland.
We wish to know the evolution of the average temperature in Switzerland, or the average temperature over a year for a given site, or some other combination of measure.

#2 Problem formulation

- Mathematics:

$$\bar{t} = \sum_i t_i(t) \cdot \Delta V_i$$

$$\bar{t} = \sum_t \sum_i t_i(t) \cdot \Delta V_i \Delta t$$

- Input: $t_i(t)$, output: $\bar{t}$

Making scientific software

#3 Algorithms description

Making scientific software

#3 Algorithms description

- Decompose program in sub-parts

Making scientific software

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts

Making scientific software

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts
- Choose data structures

#3 Algorithms description

- Decompose program in sub-parts
- Choose algorithm to solve the sub-parts
- Choose data structures
- Identify polymorphic code: class diagram

Take away message

Criterion for the projects

- Program compile and work
- Code factorization (polymorphic)
- Code documented with a short README
- Code documented with doxygen
- Code has tests