Conclusions

P. Ronchese
Dipartimento di Fisica e Astronomia “G.Galilei”

Università di Padova

“Object oriented programming and C++” course

Recommendations - 1

- Define which object has the responsibility of doing what operation: e.g. look at objects in the example "`braggPlot_v6`".
- Use only one variable for each quantity: this ensures that when it's set or modified its new content can be seen everywhere across the whole program: e.g. look at the min. and max. number of points in the Bragg curve determination.
- Avoid hardcoded constant numbers.
- Insert an explicit `return` at the end of `void` functions.
- Define any "operation" in only one "point": whenever changing an operation requires modifying the code in more than one point the chance of introducing bugs increases.
- Avoid circular-dependencies in libraries.

Recommendations - 2

- Whenever a program must perform a different operation:
  - it should be possible re-running with different parameters,
  - if not, it should be possible by re-linking with different libraries,
  - if not, it should be possible by modifying and recompiling one source file,
  - if not, it should be possible by modifying or adding several source files, while keeping the interfaces unchanged.
- Changes in the interfaces and function prototypes should occur only when major refactoring are needed.
- Try to split tasks as much as possible, aiming at:
  - avoid code replications,
  - avoid dependencies on unnecessary code/libraries.

Recommendations - 3

- Keep the code tidy:
  - split long lines,
  - insert blank lines to help identifying blocks of related operations,
  - indent properly.
- Choose a coding style and use it consistently:
  - insert or remove blanks before/after parentheses, equal signs, `if`, `for` and similar statements,
  - insert or remove new line before blocks `{}` ,
  - ...
- Document the code.