Programmiermethoden des Wissenschaftlichen Rechnens

Winter semester 2018/2019
Prof. Dr. Marc Alexander Schweitzer
Clelia Albrecht and Albert Ziegenhagel

UNIVERSITÄT BONN

Exercise sheet 4.

Exercise 28. (Regular types for linear algebra)

a) Inspect your linear algebra library from the last exercise with respect to regularity of your types. Are they already regular? If not, what is missing?

b) If your types are not yet regular, make them so.

c) What influence has the regularity of your types on the interface and implementation of operations such as addition and multiplication? Pay special attention to usability and performance. Improve your operations so that they make use of the regularity of your types. Explain what improvements you could apply to your functions because of the regularity of your types. If your types where regular already explain what properties of regularities you have used in your interfaces/implementations.

Exercise 29. (Generic types for linear algebra)

a) Make your linear algebra library generic with respect to the scalar type. Test your matrix/vector operations with at least `float`, `double` and `std::complex<double>`.

b) Think about your solver library: Why is it not so easy to make it work with complex numbers as with real numbers?

c) Make your solver library generic with respect to the scalar type. Test your solvers with at least `float` and `double`.

Exercise 30. (Revisiting B-Splines)

a) Inspect your B-Spline library from the last exercise with respect to regularity of your types. Are they already regular? If not, what is missing?

b) If your types are not yet regular, make them so.

c) What influence has the regularity of your types on the interface and implementation of your evaluation and interpolation functions? Pay special attention to usability and performance. Improve your operations so that they make use of the regularity of your types. Explain what improvements you could apply to your functions because of the regularity of your types. If your types where regular already explain what properties of regularities you have used in your interfaces/implementations.

d) Make your B-Spline library generic with respect to the scalar type and the space dimension. Test your library in 1, 2 and 3 space dimensions.

Exercise 31. (Inheritance for a hierarchy of solvers - individual hand-in)

a) Make yourself familiar with the concept of class inheritance in C++.

b) Make sure you understood:

- Public, protected and private inheritance.
- Inheritance of interfaces vs. inheritance of implementation.
- Virtual and pure-virtual functions.
- Virtual destructors (their meaning and when/why you need them).
- Abstract classes.
- The `override` specifier.

c) Write a short summary (2-3 sentences per point) about class inheritance as provided by C++.

d) Design a class hierarchy for your solvers and preconditioners. How does this improve your library with respect to:

- Usability
- Performance
- Extensibility
- Readability

e) What are potential drawbacks of using polymorphism to model your solvers?