Hands-on exercises

Modern Fortran: Meta-programming with Fypp

Modern Fortran: Useful tools and techniques

Bálint Aradi

Before your start the exercises, please download the corresponding archives containing the initial project files (especially makefiles).

Exercise 1 – handling the basics

- Write a simple mathematical library which provides a function for calculating the factorial of an integer.
- Write a unit testing framework for testing the functionality of your module.
- Check for certain factorials explicitely.
- Check for the important special cases (e.g. factorial of 0).
- Use random number based consistency checks.
- Add parameterized tests for testing for explicit numbers without having to repeat code.
- Check, whether your test ensure 100% coverage.
- Document the API of your library and extract the documentation via Doxygen or Ford.

Exercise 2 – using Fortran templates

- Write a simple template which allows you to swap two objects of arbitrary type, rank and shape (provided they have the same type, rank and shape).
- Generate a module containing special implementations for swapping two reals, two integers, two one-dimensional integer arrays and two one-dimensional real arrays.
- Write a test framework for testing the functionality of your library.
- Document the API of your library and extract the documentation via Doxygen or Ford.

Exercise 3 – writing generic mathematical packages

- Write a mathematical package which solves linear systems of equations.
- It should work with both, real and complex numbers, both in single as well as in double precisions.
- If the package is built with LAPACK support, it should call the corresponding LAPACK functions. If it is built without LAPACK, it should use a hand-coded LU-decomposition (or Gauss-elimination) instead.
- Create unit tests (explicit and randomized ones) to check the functionality of your library.

- Make sure, you reach 100% test coverage.
- Document the API of your library and extract the documentation via Doxygen or Ford.