Course

Advanced Programming
Academic Year 2023/2024

Programs

- (M.Sc.) Data Science and Artificial Intelligence @ UniTS
- (M.Sc.) Scientific and Data Intensive Computing @ UniTS
- (M.Sc.) Mathematics @ UniTS
- (Ph.D.) Mathematical Analysis, Modelling, and Applications @ SISSA

Instructor

Dr. Pasquale Claudio Africa <pafrica@sissa.it>

Goals

1. Knowledge and Understanding:
Students will acquire a comprehensive understanding of advanced programming concepts, specifically in C++ and Python. They will become familiar with object-oriented and generic programming paradigms, as well as proficient in utilizing common data structures, algorithms, and relevant libraries and frameworks for scientific computing. Furthermore, students will be introduced to fundamental software development tools in a Linux environment, encompassing essential aspects like software documentation, version control, testing, and project management.

2. Practical Application of Knowledge:
Students will be adept at applying their advanced programming knowledge to solve both familiar and novel coding challenges. They will be able to develop advanced programs in C++ and Python, choosing and designing appropriate data structures and algorithms to address complex real-world computational problems.

3. Cognitive Skills:
Students will develop critical thinking and analytical skills, enabling them to assess the effectiveness, flexibility, and efficiency of code implementations. They will be capable of evaluating different approaches to problem-solving and selecting optimal solutions based on informed analysis. By the end of the course, students will have developed the ability to design, implement, test, debug, and optimize complex scientific software.

4. Communication Skills:
Through a combination of homework assignments, code documentation, and presentations, students will learn to effectively communicate their design decisions, code rationale, and problem-solving strategies. They will be skilled at conveying technical information to both technical and non-technical audiences.

5. Autonomy and Professional Development:
Students will acquire self-directed learning abilities, enabling them to stay current with evolving and new programming languages, tools, and technologies. They will be equipped to independently identify and troubleshoot coding issues and propose innovative solutions. They will also gain experience in team-based software development practices by working on group programming projects.

Required skills

Former knowledge of programming fundamentals (syntax, data types, variables, control structures, functions) is required for this course.

While prior experience with a programming language, such as C, C++, Java, or Python, is recommended, it is not mandatory.

Subjects

The UNIX shell. Version control: Git. The build process: preprocessor, compiler, linker and loader. Introduction to GNU Make and CMake. Best practices for writing reliable code: error handling, unit testing, and software documentation.

Advanced programming in C++. Built-in data types. Variables, pointers and references. Control structures. Functions. Custom data types and classes. Object-oriented programming, inheritance and polymorphism. Lambda functions and functors. Templates and generic programming. Smart pointers. Containers and algorithms from the Standard Template Library. New features of C++14/17/20. Debugging and profiling: introduction to GDB and Valgrind.

Advanced programming in Python. Built-in data types. Variables, lists, tuples, dictionaries and sets. Control structures. Functions. Mutable and immutable data types. Plotting. Custom data types and classes. Object-oriented programming, inheritance and polymorphism. Modules and packages. Introduction to NumPy and SciPy for scientific computing. Introduction to pandas for data analysis.

Integrating C++ and Python codes.

Teaching methods

The course will utilize a combination of frontal lectures and live programming demonstrations.

Dedicated hands-on sessions will be held after each frontal lecture. During the hands-on sessions, students will have the opportunity to work on problems related to the material

covered in the frontal lectures. The solutions to these problems might be discussed, allowing students to receive feedback and gain a deeper understanding of the material.

The course will maintain a balance of approximately 50% frontal lectures and 50% hands-on sessions.

The course is designed to be highly interactive, with ample opportunities for students to ask questions and engage in discussions during both the frontal lectures and hands-on sessions.

Verification of learning

The evaluation of students in this course will comprise of two components: homework assignments and a computer-based written exam.

Throughout the course, students will be assigned a series of small homework projects to complete either individually or in groups. Students are expected to submit the solution code and deliver a brief presentation, supported by slides, outlining their proposed solution and design choices.

The computer-based written exam will consist of programming questions and exercises designed to assess the students' mastery of the course material. Some of the exam solutions must be submitted in digital format rather than in handwritten form.

Additionally, an optional oral exam may be requested by either the student or the instructor. The oral exam may increase or decrease the student's grade by up to 3 points.

The maximum achievable grade is 30. Honors may be granted in exceptional cases, provided the final grade exceeds 30.

Overall, the evaluation process is designed to encourage students to actively engage with the material and to develop practical skills that will be useful in their future careers.

Books and material

The instructor will provide support material and references throughout the course. In addition, there are many free online resources available to supplement the course material.

For students who prefer to read books, the following references are recommended:

[1] Programming - Principles and Practice Using C++, Bjarne Stroustrup, Addison-Wesley, May 2014 https://www.stroustrup.com/programming.html

[2] Learning scientific programming with Python, Christian Hill, Cambridge University Press, October 2020 https://scipython.com/

These books provide in-depth coverage of the course material and can serve as valuable resources for further study.

Sustainable Development Goals - Agenda 2030

Goal 9 - Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

Goal 12 - Ensure sustainable consumption and production patterns

Extra info

To participate in this course, students will be requested to bring their own laptop equipped with a working Linux or Unix environment, whether standalone or virtualized. Students are expected to utilize either a text editor, such as Emacs, Vim, or Nano, or an Integrated Development Environment (IDE), such as Visual Studio, Eclipse, or Code::Blocks, according to their preference.

To ensure that their environment is suitable for the course, students should ensure that it meets the following requirements:

- The environment must have a C++ compiler installed with full support for C++17, such as GCC 10 or newer, or Clang 11 or newer.
- The environment must have Python 3 installed.
- The environment must meet the minimum system requirements for running a Linux or Unix environment with the necessary software and tools for the course.

Any recent Linux distribution, such as Ubuntu 22.04 or newer, or Debian 11 or newer, or macOS system that meets these requirements should be suitable for the course.