COMPUTER PROGRAMMING 2

Bachelor in Computer Science and Artificial Intelligence
BCSAI SEP-2024 CP2-CSAI.3.M.A

Area Computer Science and AI

Number of sessions: 30

Academic year: 24-25

Degree course: THIRD

Number of credits: 6.0

Semester: 1º

Category: COMPULSORY

Language: English

Professor: **RAÚL PÉREZ PELAEZ**

E-mail: rperezp@faculty.ie.edu

Raul P. Pelaez did his PhD on high-performance simulation of complex fluids at the Universidad Autonoma de Madrid in Rafael Delgado-Buscalioni's group, in close collaboration with Aleksandard Donev's group (Courant Institue, New York University). During this time he developed Universally Adaptable Multiscale Molecular Dynamics (UAMMD), a multipurpose, modular GPU software package for nano and microhydrodynamics of suspensions (uammd.readthedocs.io).

He is currently a Postdoctoral researcher at the Universitat Pompeu Fabra (Barcelona). RPP acts as a machine learning research director at OpenMM, a popular high-performance, customizable molecular mechanics software package. In this role, he researches high-performance approaches for machine learning-enabled forcefields (neural network potentials) for biomolecular simulations.

Office Hours

Office hours will be on request. Please contact at:

rperezp@faculty.ie.edu

SUBJECT DESCRIPTION

C++ remains crucial in performance-critical domains such as game development, systems programming, high-frequency trading, and embedded systems, where its low-level control, efficient memory management, and ability to interface directly with hardware make it indispensable for creating fast, resource-efficient applications.

This comprehensive C++ course offers an in-depth exploration of modern C++ programming. The curriculum covers fundamental concepts, advanced features, and best practices in software engineering. Students will progress from basic syntax and object-oriented programming principles to advanced topics such as templates, STL, and memory management. The course emphasizes practical skills through hands-on coding and good software engineering practices using real-world tools and techniques for debugging, testing, version control, documentation and build systems. By the end of the course, students will have developed a strong foundation in C++ programming, enabling them to design and implement efficient, robust software solutions. The course includes continuous individual presentations (small homework examples) and a group project, providing opportunities to apply learned concepts to real-world scenarios.

LEARNING OBJECTIVES

By the end of this course students should be able to:

1. Understand the basic programming features and syntax of the C++ language.
2. Understand the basic memory concepts in C++ (values, pointers, references).
3. Understand and apply object-oriented programming principles in C++.
4. Implement fundamental software engineering practices, including version control and unit testing.
5. Manage memory effectively in C++ programs.
6. Create programs in C++ that are correct, robust and capable of being understood, reused and modified by others.

TEACHING METHODOLOGY

IE University teaching method is defined by its collaborative, active, and applied nature. Students actively participate in the whole process to build their knowledge and sharpen their skills. Professor's main role is to lead and guide students to achieve the learning objectives of the course. This is done by engaging in a diverse range of teaching techniques and different types of learning activities such as the following:

Learning Activity	Weighting	Estimated time a student should dedicate to prepare for and participate in
Lectures	40.0 %	60.0 hours
Group work	26.7 %	40.0 hours
Individual studying	33.3 %	50.0 hours
TOTAL	100.0 %	150.0 hours

AI POLICY

In this course, the use of generative artificial intelligence (GenAI) is encouraged, with the goal of developing an informed critical perspective on potential uses and generated outputs.

However, be aware of the limits of GenAI in its current state of development:

- If you provide minimum effort prompts, you will get low quality results. You will need to refine your prompts to get good outcomes. This will take work.

- Don't take ChatGPT's or any GenAI's output at face value. Assume it is wrong unless you either know the answer or can cross-check it with another source. You are responsible for any errors or omissions. You will be able to validate the outputs of GenAI for topics you understand.
- LLMs are a good productivity/search tool when you are an expert on the field and can thus be critical of its output. Be aware of this when using it for learning.
- AI is a tool, but one that you need to acknowledge using. Failure to do so is in violation of academic honesty policies. Acknowledging the use of AI will not impact your grade.
- Coding LLMs (Github copilot) are really good at writing boiler-plate code (which C++ has lots of). This is an incredible time-saver when one understands the produced code. Be aware and critical of the impact generating this kind of code has for your learning process.

PROGRAM

The following description of the material covered is tentative. An attempt will be made to cover all listed topics and to include other advanced topics that will help the student throughout their career in computer science. However, the pace of the classes will depend on group performance, which may introduce some variations in the syllabus.

SESSION 1 (LIVE IN-PERSON)

Course Overview and Introduction to C++

- Course objectives and structure
- History and evolution of C++, C++ Standards/dialects, differences with C.
- Why/where C++ is important
- Comparison with other programming languages
- Overview of the C++ Standard Library and other language resources

SESSION 2 (LIVE IN-PERSON)

Quiz session on Basic Programming Syntax in C

Going over examples with increasing complexity, covering:

- Variables, data types, and basic operators. Type deduction (auto)
- Control structures: if, else, switch, loops (for, while, do-while)
- Functions: declaration, definition, and scope
- Basic input/output operations: cin, cout
- First programs, godbolt.

SESSION 3 (LIVE IN-PERSON)

Version Control with git and Github

Git, github, pull requests, documentation.

SESSION 4 (LIVE IN-PERSON)

Unit Testing

- Writing unit tests with Google Test

- Best practices for testing
- Test-driven development

SESSION 5 (LIVE IN-PERSON)

Namespaces, Inline Functions, Preprocessor, Directives and Macros

- Namespaces, scope.
- Inline functions and their advantages
- Introduction to the preprocessor,
- Common preprocessor directives: #include, #define, #if, #endif
- Macros and their applications, conditional compilation

SESSION 6 (LIVE IN-PERSON)

Memory model, containers, pointers and references

- Memory, addresses and pointers
- Arrays, access to elements
- References, alternatives to pointer use

SESSION 7 (LIVE IN-PERSON)

Debugging Techniques

- Debugging tools and techniques
- Using gdb for debugging
- Common debugging scenarios

SESSION 8 (LIVE IN-PERSON)

Software Delivery

- Compilation strategies: make, CMake
- Code Packaging: pip, conda, github releases…

SESSION 9 (LIVE IN-PERSON)

Classes and Objects

- Defining classes and creating objects
- Access specifiers: public, private, protected
- Member functions and data members
- Nested classes
- The "this" pointer
- The “const”,“static” and “friend” keywords

SESSION 10 (LIVE IN-PERSON)

Constructors and Destructors, other special methods

- Purpose of constructors and destructors
- Types of constructors: default, parameterized, copy

- Rules of 3 and 5
- Destructor implementation and usage
- Operators.

SESSION 11 (LIVE IN-PERSON)

Inheritance & Polymorphism
- Base and derived classes
- Types of inheritance: single, multiple, multilevel, hierarchical
- Overriding base class methods
- Constructor and destructor invocation order
- Function overloadingVirtual functions and runtime polymorphism
- Pure virtual functions and abstract classes
- Polymorphic behavior with pointers and references, static_cast and dynamic_cast
- User Defined Type conversion
- Runtime casting

SESSION 12 (LIVE IN-PERSON)

Good practices for Object Oriented code
- Principles of OO design
- Common pitfalls: Classitis, over-application of design patterns.
- The expression problem
- Deep and shallow modules
- Design it thrice
- Interface and implementation inheritance

SESSION 13 (LIVE IN-PERSON)

Advance memory handling
- Dynamic memory allocation using new and delete
- Memory leaks and how to avoid them
- Smart pointers: unique_ptr, shared_ptr, weak_ptr
- std::vector.
- RAII (Resource Acquisition Is Initialization)

SESSION 14 (LIVE IN-PERSON)

Files
- File stream classes: ifstream, ofstream, fstream
- Opening and closing files
- Reading from and writing to files
- Text vs. binary mode
- File pointers and seeking
- Error handling in file I/O

SESSION 15 (LIVE IN-PERSON)

Function Templates

- Template syntax and definition
- Using function templates
- Template specialization

SESSION 16 (LIVE IN-PERSON)

Class Templates

- Template class definition
- Instantiation of template classes
- Non-type template parameters

SESSION 17 (LIVE IN-PERSON)

STL Containers

- Overview of STL
- Sequential containers: vector, list, deque
- Associative containers: set, map, multiset, multimap
- Container adapters: stack, queue, priority_queue

SESSION 18 (LIVE IN-PERSON)

STL Iterators and Algorithms

- Types of iterators: input, output, forward, bidirectional, random access
- Common iterator functions
- STL algorithms: sorting, searching, and modifying sequences
- Lambda expressions with STL algorithms

SESSION 19 (LIVE IN-PERSON)

Recovering from errors

- Exception handling model: try, catch, throw
- Standard exceptions and creating custom exceptions
- Stack unwinding and rethrowing exceptions
- Writing exception-safe code

SESSION 20 (LIVE IN-PERSON)

Technicalities

- Enumerations, unions…
- Order of evaluation, comma-operator
- Declarations and definitions
- Other keywords (extern, volatile)

SESSION 21 (LIVE IN-PERSON)

What we learned so far
- Recap of the course until this moment
- Group Assignment Exploration (team members + topic)

SESSION 22 (LIVE IN-PERSON)

Midterm - Intermediate Test

SESSION 23 (LIVE IN-PERSON)

Profiling and identifying bottlenecks
- Code optimization: inlining, loop unrolling
- Memory optimization: cache optimization, avoiding fragmentation

SESSION 24 (LIVE IN-PERSON)

Parallel computing primer
- tbb
- openmp
- std::async and std::future
- GPU computing with thrust

SESSION 25 (LIVE IN-PERSON)

Recap, Group Assignment Support

SESSION 26 (LIVE IN-PERSON)

Recap, Group Assignment Support

SESSION 27 (LIVE IN-PERSON)

Recap, Group Assignment Support

SESSION 28 (LIVE IN-PERSON)

Group Assignment presentations

SESSION 29 (LIVE IN-PERSON)

Recap + Problems

SESSION 30 (LIVE IN-PERSON)

Final Exam

EVALUATION CRITERIA

criteria	percentage	Learning Objectives	Comments
Final Exam	25 %		
Individual presentations	20 %		Homework completion and short presentations of it
Group Project	20 %		Includes a presentation
Class Participation	10 %		
Intermediate tests	25 %		

RE-SIT / RE-TAKE POLICY

Each student has four chances to pass any given course distributed over two consecutive academic years: ordinary call exams and extraordinary call exams (re-sits) in June/July. Students who do not comply with the 80% attendance rule during the semester will fail both calls for this Academic Year (ordinary and extraordinary) and have to re-take the course (i.e., re-enroll) in the next Academic Year. Evaluation criteria: Students failing the course in the ordinary call (during the semester) will have to re-sit the exam in June / July (except those not complying with the attendance rule, who will not have that opportunity and must directly re-enroll in the course on the next Academic Year). The extraordinary call exams in June / July (re-sits) require your physical presence at the campus you are enrolled in (Segovia or Madrid). There is no possibility to change the date, location or format of any exam, under any circumstances. Dates and location of the June / July re-sit exams will be posted in advance. Please consider this when planning your summer. The June / July re-sit exam will consist of a comprehensive exam. Your final grade for the course will depend on your performance in this exam only; continuous evaluation over the semester will not be taken into consideration. Students will have to achieve the minimum passing grade of 5 and can obtain a maximum grade of 8.0 (out of 10.0) – i.e., “notable” in the re-sit exam.

Retakers: Students who failed the subject on a previous Academic Year and are now re-enrolled as re-takers in a course will be needed to check the syllabus of the assigned professor, as well as contact the professor individually, regarding the specific evaluation criteria for them as retakers in the course during that semester (ordinary call of that Academic Year). The maximum grade that may be obtained in the retake exam (3rd call) is 10.0. After the professor grades ordinary and extraordinary call exams, you will have the possibility to attend a review session for that exam and course grade. Please be available to attend the session in order to clarify any concerns you might have regarding your exam. Your professor will inform you about the time and place of the review session. Any grade appeals require that the student attended the review session prior to appealing. Students failing more than 18 ECTS credits after the June / July re-sits will be asked to leave the Program. Please, make sure to prepare yourself well for the exams in order to pass your failed subjects. In case you decide to skip the opportunity to re-sit for an exam during the June/July extraordinary call, you will need to enroll in that course again for the next Academic Year as a re-taker and pay the corresponding extra cost. As you know, students have a total of four allowed calls to pass a given subject or course, in order to remain in the program

BIBLIOGRAPHY

Recommended

- Bjarne Stroustrup. (2024). *Programming: Principles and Practice Using C++.* third edition.. ISBN 9780138308681 (Digital)
https://stroustrup.com/programming.html
- Bjarne Stroustrup. (2022). *A Tour of C++.* third edition. ISBN 0136816487 (Digital)

https://stroustrup.com/tour3.html

BEHAVIOR RULES

Please, check the University's Code of Conduct here. The Program Director may provide further indications.

ATTENDANCE POLICY

Please, check the University's Attendance Policy here. The Program Director may provide further indications.

ETHICAL POLICY

Please, check the University's Ethics Code here. The Program Director may provide further indications.