Programming

Academic Year: (2020 / 2021) **Review date: 10/07/2020 16:57:55**

Department assigned to the subject: Computer Science and Engineering Department

Coordinating teacher: GARCIA SANCHEZ, JOSE DANIEL

Type: Compulsory ECTS Credits : 6.0

Year : 1 Semester : 1

OBJECTIVES

Skills:
* To be able to apply acquired knowledge and their ability to solve problems in new or unknown environments within wider (or multidisciplinary) contexts related with their subject matter.
* To get learning skills that allow to continue studying in a way that should be mostly self-drive or autonomous
* Ability to understand and apply methods and techniques in the scope of Computer Engineering for Financial markets.

Learning outcomes:
* To be aware of main programming languages used for financial software development.
* Ability to implement software for financial sector.
* Knowledge on high performance programming.

DESCRIPTION OF CONTENTS: PROGRAMME

1. High Performance Computing Fundamentals.
1.1. Performance problema.
1.2. Limits of performance. Performance laws.
1.3. High performance in sequential applications.
1.4. Parallelism and high performance computing. Kinds of parallelism.

2. Programming languages
2.1. Programing languages and performance perspective.
2.2. Introduction to C++.
2.3. Interface Inheritance versus implementation inheritance.
2.4. Dynamic polymorphism.

2. High Performance Computing Fundamentals.
2.1. Performance problem.
2.2. Limits of performance. Performance laws.
2.3. High performance in sequential applications.
2.4. Parallelism and high performance computing. Kinds of parallelism.

3. Memory management.
3.1. Automatic and static storage.
3.2. The free store.
3.3. Parameters passing and result return by value and reference.
3.4. Move semantics.
3.5. Smart pointers.

4. Generic programming
4.1. Templates and generic programming. Static polymorphism.
4.2. Concepts and generic programming
4.3. Function objects and lambda expressions.

4.4. Variadic templates.
4.5. STL architecture.
4.6. STL: Containers and iterators.
4.7. STL: Algorithms.

5. Libraries and interoperability.
5.1. Library kinds: static and dynamic.
5.2. Header-only libraries.
5.3. Includes and compilation time. Forward declarations. Optimizing includes.
5.4. Libraries organizations and namespaces.
5.5. Build engines.
5.6. Languages interoperability.
5.7. Overview of frequently used libraries.

6. Code optimization.
6.1. Code optimizer: effects and limits.
6.2. Code vectorization: Limits of automated vectorization.
6.3. Supporting code hand vectorization.
6.4. Function inline expansion: myths and realities.
6.5. Elimination of temporal objects.
6.6. Compile-time Expression evaluation.
6.7. Stack unwiding: the noexcept specifier.

7. Applications performance analysis.
7.1. Performance analysis mechanisms.
7.2. Intrusive code analyisis.
7.3. Application profiling.
7.4. Hardware and system counters analysis.
7.5. Processor cache impact: Analysis.
7.6. Performance analysis in multithreaded code.

8. Threads and frameworks for concurrency and parallelism.
8.1. Concurrent programming in C++.
8.2. Threads, mutex and condition variable.
8.3. Futures and packaged tasks.
8.4. Shared memory parallel programming alternatives.
8.5. Standard parallel programming: language solutions versus library solutions.
8.6. Parallelism frameworks.

LEARNING ACTIVITIES AND METHODOLOGY

Activities:
* Theoretical Classes: Theoretical presentations accompanied by electronic material such as digital presentations.
* Theoretical Practical classes: Combination of lectures accompanied by the resolution of practical exercises.
* Laboratory practices: Practices to develop in specific laboratories for different subjects.
* Tutoring: Personal student tutoring
* E-learning activities: Course fora, discussion fora and other training e-learning activities.
* Individual student work: individual student activities that complement the other activities and exam preparation.

Teaching methodology
* Teacher class presentations with computer and audiovisual support, in which the main course concepts are developed and bibliography is provided to supplement student learning.
* Resolution of practical cases, problems, etc. posed by the teacher individually or in group
* Expositions and class discussion under the moderation of teacher related to the course content and practical cases.
* Elaboration of reports individually or in group
* Specific e-learning activities, including viewing recorded content, self-correcting activities, participation in fora, and any other mechanism online teaching.

The practical component in this course assingments will be developed for applications performance improvement. Those assignments may include, among others, applications performance analysis, performance problems identification, code optimization, and parallelization techniques.
For these more practical content, attendance to laboratories and individual or group work outside the classroom may be combined with tracking and tutoring students through fora and other mechanisms for discussion. Other e-learning strategies will be also used, such as self-assessment of the work done, all supported through Aula Global.

ASSESSMENT SYSTEM

% end-of-term-examination/test:	30
% of continuous assessment (assigments, laboratory, practicals…):	70

Individual or group assignments performed througout the term: 70%
Final exam: 30%

BASIC BIBLIOGRAPHY

- Anthony Williams C++ Concurrency in Action. Practical Multithreading. 2nd Edition, Manning, 2019

- Bjarne Stroustrup Programming - Principles and Practice. 2nd Edition, Addison-Wesley, 2014

- Michael Voss, Rafael Asenjo, James Reinders Pro TBB: C++ Parallel Programming with Threading Building Blocks, APress, 2019

ADDITIONAL BIBLIOGRAPHY

- Bjarne Stroustrup The C++ Programming Language. 4th Edition, Addison-Wesley, 2013

- Bjarne Stroustrup A Tour of C++. 2nd Edition., Addison-Wesley, 2018

- Gerassimos Barlas Multicore and GPU Programming: An Integrated Approach, Morgan-Kaufmann, 2014

- Kurt Guntheroth Optimized C++, O'Reilly, 2016

- Peter Gottschling Discovering Modern C++: An intensive course for scientists, engineers and programmers, Addison-Wesley, 2015

BASIC ELECTRONIC RESOURCES

- CppReference . C++ Reference: https://en.cppreference.com/w/