Object-Oriented Programming

1. **Course number and name:** 020CPPES1 Object-Oriented Programming

2. **Credits and contact hours:** 6 ECTS credits, 3x1:15 course hours + 1:15 lab hours

3. **Name(s) of instructor(s) or course coordinator(s):** Youssef El Bakouny (coordinator), Chantal Saad Hajjar, Ahmad Audi

4. **Instructional materials:** PowerPoint slides; Moodle ressources; lab sheets, assignments, Massive Online Open Courses (MOOCs) on Coursera: *Initiation à la programmation (en C++)*, *Introduction à la programmation orientée objet (en C++)* by Jean-Cédric Chappelier and Jamila Sam.

5. **Specific course information**
   a. **Catalog description:**
      C/C++ syntax: typed variable declarations, basic I/O, expressions, implicit and explicit type conversion, conditional branching, for and while loops, functions and prototypes, parameter passing and overloading. Arrays, strings, cyclic dependency resolution, references, pointers and manual memory management. Deep copy and smart pointers. The object-oriented paradigm: abstraction, encapsulation, inheritance and polymorphism. Definition of classes, constructors, destructors, attributes, methods, the "static" keyword, access modifiers and operator overloading. Development environment with VS Code. Compiling with CMake. Code versioning with git and github.

   b. **Prerequisites:** 020IF2NI3 Programming 2 or 020IF2CI3 Programming 2

   c. **Required** for CCE and EE students

6. **Educational objectives for the course**
   a. **Specific outcomes of instruction:**
      – Implement a procedural C program.
      – Implement an object-oriented C++ program in congruence with current best practices with regards to correctness, maintainability and performance.
      – Analyze a C++ program with regards to its compliance to a given specification.
      – Design and implement a C++ program to solve a complex problem.
      – Evaluate the maintainability of C++ code and propose appropriate improvements.

b. **PI addressed by the course:**

PI	1.2	1.3	2.3	2.4	2.5	6.4	7.1
Covered	x	x	x	x	x	x	x
Assessed	x	x	x		x	x	

7. **Brief list of topics to be covered**
   – Introduction and History (1 lecture)
   – Start by compiling a "cpp" file using a command-line based compiler and then proceed unto using Visual Studio Code instead which integrates the compilation next to the coding panel (1 lecture)
   – Typed variable declarations, basic I/O, expressions (2 lectures)
   – Implicit and explicit type conversion, conditional branching, for and while loops (2 lectures)
   – Lab 1 : C/C++ Basics
   Setting up a simple Visual Studio Code development environment for resolving simple exercices Code versioning: using a local Git repo, pushing/pulling commits to/from GitHub, GitHub Actions for CI/CD (2 sessions)
   – functions and prototypes, parameter passing, overloading (2 lectures)
   – std::vector, std::string, C array, C string, multidimensional arrays, typedef, pointers and references (3 lectures)
   – Lab 2 : Functions, arrays and stdlib (2 sessions)
   – Manual memory management: declaration of references, pointers and dynamic memory allocation, smart pointers, C alloc / free and C++ new / delete, common segmentation fault errors (example: never return a reference to objects allocated on the stack) (3 lectures)
   – Lab 3: Manual memory management (2 sessions)
   – Structures (1 lecture)
   – Introduction to Object-Oriented Programming (OOP) Abstraction and Encapsulation Classes and Instances (1 lecture)
   – Private and public keywords, constructors and destructors, methods: actions and predicates, static attributes and methods (2 lectures)
   – Manual memory management and OOP: deep copy and move semantics (3 lectures)
   – Operator overloading: internal/external overloads and overload levels Reuse of constructor and destructor code during the implementation of "operator=" (4 lectures)
   – Using CMake to automate C++ project building under Visual Studio Code (1 lecture)
   – Lab 4: Classes (2 sessions)
   – Inheritance: concept, static binding and shadowing, protected keyword, constructors and destructors, shadowing, overloading and overriding (2 lectures)
   – constructor / destructor invocation order, virtual destructors, upcast and downcast, abstract classes (1 lecture)
   – Polymorphism: static and dynamic types, dynamic binding and virtual methods, method overriding (2 lectures)
   – Lab 5: Inheritance and Polymorphism (2 sessions)

- Heterogeneous collections: abstract classes, deep copy of heterogeneous collections, the override keyword (2 lectures)
- Templates (2 lectures)
- Exception handling and inheritance (1 lecture)
- Multiple inheritance: dangers and the virtual class solution (1 lecture)
- Lab 6: Heterogeneous collections (2 lectures)