



DISCIPLINE ACADEMIC SHEET
ACADEMIC YEAR 2025 – 2026

1. PROGRAMME DATA

1.1. Higher Education Institution	UNIVERSITY OF CRAIOVA
1.2. Faculty	Automation, Computers and Electronics
1.3. Department	Computers and Information Technology
1.4. Field of Study	Computers and Information Technology
1.5. Study level	L(License / Undergraduate)
1.6. Organizational form	IF
1.7. Study Program (name/code) /Qualification	Computers / L2060101010

2. DISCIPLINE DATA

2.1 Discipline Name	Object Oriented Programming						
2.2 Course Activities Holder							
2.3. Practical Activities Holder							
2.4. Study Year	II	2.5. Semester	1	2.6. Evaluation type	E	2.7. Discipline regime	DOB

3. ESTIMATED TOTAL TIME (hours per semester of teaching activities)

3.1. Number of hours per week	5	in which: 3.2 course	3	3.3. sem./lab./pr.	2
3.4. Total hours of curriculum	70	in which: 3.5 course	42	3.6. sem./lab./pr.	28
3.7. Time distribution - hours/week					hours
Study after manual, course support, bibliography and notes					15
Additional documentation in library, on specialized electronic platforms and field					20
Training seminars / labs, homework, portfolios and essays					15
Tutorial					
Examinations					
Other activities: consultations, student meetings					5
Total hours per individual activities					55
3.8. Total hours per semester					125
3.9. Number of credits					5

4. PRECONDITIONS (where appropriate)

4.1. of curriculum	- The students must have knowledge of algorithms - Good knowledge of C programming language
4.2. of competence	English language skills

5. CONDITIONS (where appropriate)

5.1. of the course	Course lectures are taught using the video projector + interactive activities (discussions with students).
--------------------	--

	Students are provided with course materials and up-to-date documentation (in electronic format).
5.2. of seminar/ laboratory/project	The applicative activities are carried out using tutorials and practical demonstrations; students are provided with up-to-date educational materials and continuous feedback.

6. COURSE OBJECTIVES – expected learning outcomes that are supported by completing and passing the course

Knowledge	Student/Graduate: 1. Core Paradigm & Concepts: Explain and apply fundamental object-oriented programming concepts—including data encapsulation, abstraction, class hierarchies, inheritance, and compile-time/runtime polymorphism—as well as their underlying memory models and runtime behavior. 2. Language-Specific Architecture & Idioms: Identify and describe advanced C++ language mechanisms and modern idioms, explicit memory management with smart pointers, operator overloading, template metaprogramming, and exception safety guarantees. 3. Design Principles & Software Engineering Practices: Recognize and analyze modular software architecture principles (including S from SOLID design criteria, unified modeling notation concepts, and foundational software design patterns) to evaluate code maintainability, reusability, and structural decoupling.
Skills (Abilities)	Student/Graduate: 1. Implementation & Program Construction: Design, code, debug, and refactor robust, modular applications in C++ by translating high-level structural requirements into clean class interfaces, leveraging inheritance, virtual methods, dynamic memory allocation (smart pointers), and standard library algorithms (STL). 2. Architectural Design & Pattern Application: Apply S from SOLID design principles to decouple software components, eliminate code duplication, and improve system maintainability. 3. Tooling, Testing & Verification: Utilize professional development tools—including modern C++ IDEs, version control systems (Git), static analyzers, and unit testing frameworks—to build, test, trace, and resolve runtime exceptions and dynamic memory leaks in object-oriented programs.
Responsibility and autonomy	Student/Graduate: 1. Independent Problem-Solving & Technical Self-Direction: Demonstrate the ability to independently analyze software specifications, research technical documentation and C++ language standards, and synthesize modular solutions to complex algorithmic and structural programming problems without continuous supervision. 2. Collaborative Software Development & Professional Ethics: Collaborate effectively within small software development teams using industry-standard version control workflows (e.g., Git), adhering to professional coding standards, peer code review practices, and intellectual property/academic integrity ethics. 3. Continuous Professional Development & Adaptive Learning: Critically evaluate individual code quality and software architectures, systematically identify and resolve bugs or memory management flaws, and demonstrate self-directed learning in adopting modern software design paradigms and emerging C++ standards.

7. CONTENT

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction. Modular Programming. C extensions	face to face	- Teaching using video projector. - Black board is used for explanations. - Course in pdf format	6
2. Defining and using classes	face to face		6
3. Class Composition	face to face		6
4. Inheritance. Class hierarchy	face to face		6
5. Overloading methods and operators	face to face		3
6. Friend, Forward and Nested classes	face to face		3
7. Template classes	face to face		3
8. Polymorphism	face to face		6
9. Object Oriented Programming overview	face to face		3
Bibliography:			
1. Design Patterns: Elements of Reusable Object-Oriented SoftwareErich Gamma, Richard Helm, Ralph Johnson, John Vlissides			
2. Object-Oriented Software Construction Bertrand Meyer			
3. Object-Oriented Analysis and Design with Applications Grady Booch et al.			
4. Head First Object-Oriented Analysis and Design Brett D. McLaughlin, Gary Pollice, David West			

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
1. Modular Programming	face to face	Explanations, practical demonstration, example	4
2. C extensions	face to face		4
3. Class Composition	face to face		4
4. Inheritance. Class hierarchy	face to face		4
5. Overloading methods and operators	face to face		2
6. Friend, Forward and Nested classes	face to face		2
7. Template classes	face to face		2
8. Polymorphism	face to face		4
9. Object Oriented Programming overview	face to face		2
Bibliography:			
1. Object-Oriented Programming in C++ (Includes Lab Exercises) Robert Lafore			
2. Hands-On C++: Laboratory Exercises in Object-Oriented Programming Joel C. Adams			
3. Starting Out with C++: From Control Structures through Objects Tony Gaddis			
4. C++ Programming: Program Design Including Data Structures D.S. Malik			
5. Problem Solving, Abstraction, and Design Using C++ Frank L. Friedman & Elliot B. Koffman			

8. Correlating the content of the discipline with the expectations of representatives from the epistemic community, professional associations and representative employers in the field related to the program

The content of this discipline is continuously aligned with international academic standards, industry requirements, and recommendations from professional bodies.
The topics and practical laboratory themes are periodically validated against feedback from industry partners, internship evaluations, and advisory board meetings with local IT employers to maintain alignment with current technological trends.

9. EVALUATION

Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
9.4. Course	Good knowledge of OOP concepts. Ability to use and apply OOP concepts	Examination paper. Course activity and exercises	40% Examination Paper + 20% Course Activity/Exercises + 40% Laboratory Grade
9.5. Seminar/laboratory/project	Lab: Good knowledge of OOP concepts. Ability to use and apply OOP concepts	Solving homework or project, as discussed with teacher	100%
9.6. Minimum standard of performance			
C extensions, Class Composition, Inheritance. Class hierarchy, Overloading methods and operators			

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
23.08.2025		

Date of Approval in the Department	Signature of the Department Director,
25.08.2025	Ș.I.dr.ing. Nicolae ENESCU

Date of Approval in the Faculty	Signature of the Dean,
	Prof.dr.ing. Dorin ȘENDRESCU