Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti
Facultatea de Electronică, Telecomunicaţii şi
Tehnologia Informaţiei

COURSE DESCRIPTION

1. Program identification information

1.1 Higher education institution	**National University of Science and Technology Politehnica Bucharest**
1.2 Faculty	**Electronics, Telecommunications and Information Technology**
1.3 Department	**Electronic Devices, Circuits and Architectures**
1.4 Domain of studies	Electronic Engineering, Telecommunications and Information Technology
1.5 Cycle of studies	Bachelor/Undergraduate
1.6 Programme of studies	Applied Electronics

2. Date despre disciplină

2.1 Course name (ro) (en)				Programarea calculatoarelor şi limbaje de programare 2 Computer Programming and Programming Languages 2			
2.2 Course Lecturer				Conf. Dr. Radu Hobincu			
2.3 Instructor for practical activities				Conf. Dr. Radu Hobincu			
2.4 Year of studies	1	2.5 Semester	II	2.6. Evaluation type	E	2.7 Course regime	Ob
2.8 Course type		F	2.9 Course code	04.F.02.O.011		2.10 Tipul de notare	Nota

3. Total estimated time (hours per semester for academic activities)

3.1 Number of hours per week	3	Out of which: 3.2 course	2.00	3.3 seminary/laboratory	1
3.4 Total hours in the curricula	42.00	Out of which: 3.5 course	28	3.6 seminary/laboratory	14
Distribution of time:					hours
Study according to the manual, course support, bibliography and hand notes Supplemental documentation (library, electronic access resources, in the field, etc) Preparation for practical activities, homework, essays, portfolios, etc.					56
Tutoring					2
Examinations					0
Other activities (if any):					0

3.7 Total hours of individual study	58.00
3.8 Total hours per semester	100
3.9 Number of ECTS credit points	4

4. Prerequisites (if applicable) (where applicable)

4.1 Curriculum	Computer Programming and Programming Languages 1.

4.2 Results of learning	Logical, structured thinking. Knowledge of C syntax.

5. Necessary conditions for the optimal development of teaching activities (where applicable)

5.1 Course	Lecture hall with computer projector.
5.2 Seminary/ Laboratory/Project	Computer laboratory, running a Linux flavored operating system, with a C++ compiler (g++), code editor and development tools (a Git client, GNU Debugger and the Valgrind profiler), prefferably also a modern IDE (Jetbrains CLion). Lab capacity should be of minimum 15 seats and should have Internet access.

6. General objective *(Reffering to the teachers' intentions for students and to what the students will be thought during the course. It offers an idea on the position of course in the scientific domain, as well as the role it has for the study programme. The course topics, the justification of including the course in the currcula of the study programme, etc. will be described in a general manner)*

The overall objective of the course is the study of a new programming paradigm: object orientation as well as the ways in which software applications can be developed using these patterns, by implementing them with the help of classes and objects. Thus, the aim is the accumulation of knowledge in the field of object-oriented software architecture and the assimilation of a new programming language: C++. In addition, the proposed examples and themes are useful for assimilating the C++ API, which in turn allows the rapid and efficient development of high-performance software solutions.

7. Competences *(Proven capacity to use knowledge, aptitudes and personal, social and/or methodological abilities in work or study situations and for personal and proffesional growth. They refflect the empolyers requirements.)*

Specific Competences	C3 Applying knowledge, concepts and basic methods to computer architecture, microcontrollers, and programming languages and techniques.
Transversal (General) Competences	CT1 Methodic analysis of the possible problems, identifying the already available solutions, when possible, to accomplish the professional obligations. CT3 Adaptation to new technologies and professional and personal development by long-life learning using printed documentation, specialized software, and electronic resources both in Romanian and in an international language.

8. Learning outcomes *(Synthetic descriptions for what a student will be capable of doing or showing at the completion of a course. The learning outcomes reflect the student's acomplishments and to a lesser extent the teachers' intentions. The learning outcomes inform the students of what is expected from them with respect to performance and to obtain the desired grades and ECTS points. They are defined in concise terms, using verbs similar to the examples below and indicate what will be required for evaluation. The learning outcomes will be formulated so that the correlation with the competences defined in section 7 is highlighted.)*

Knowledge	*The result of knowledge aquisition through learning. The knowledge represents the totality of facts, priciples, theories and practices for a given work or study field. They can be theoretical and/or factual.* Knowledge of C++ language syntax and implementation of class-based applications.

Skills	*The capacity to apply the knowledge and use the know-how for completing tasks and solving problems. The skills are described as being cognitive (requiring the use of logical, intuitive and creative thinking) or practical (implying manual dexterity and the use of methods, materials, tools and intrumentation).* Skills related to analyzing a given problem in natural language and developing an own solution to solve the problem, as well as the ability to implement this solution in the C++ language, structuring code according to the object-oriented programming paradigm.
Responsability and autonomy	*The student's capacity to autonomously and responsably apply their knowledge and skills.* Ability to search for answers and online documentation related to the syntax and API of the C++ language, as well as to search for solutions to specific problems.

9. Teaching techniques *(Student centric techniques will be considered. The means for students to participate in defining their own study path, the identification of eventual fallbacks and the remedial measures that will be adopted in those cases will be described.)*

Theoretical concepts are presented using a tablet and stylus and projected on the white screen. At the end of the lecture, the notes are exported as PDF and published on the Moodle course page. After the theoretical presentation of the necessary notions, they are demonstrated with short code samples, written, compiled and executed in real time, the whole process being displayed on the projection screen. Students are constantly asked for feedback and to contribute to the implemented solutions. During the laboratory, students are given access to a computer and a set of exercises which they must solve with help from the lab assistant and the given documentation. They are graded depending on how many problems they solve until the end of the allocated time slot.

10. Contents

COURSE		
Chapter	**Content**	**No. hours**
1	Recap - pointers and structures	2
2	Introduction in object oriented programming - classes, methods, fields and objects	2
3	Constructors, overloading and encapsulation (public/private)	2
4	Destructors, operator overloading and references	4
5	Const-qualified methods, copy-constructor, copy-assignment operator, the rule of three	2
6	Class templates and namespaces	4
7	Inheritance and protected access modifier. Virtual, pure virtual methods, abstract classes and polymorphism	4
8	Exception handling	2
9	Regular expressions	2
10	Introduction in parallel programming - std::thread and std::future. Barriers - std::mutex	2
11	Semaphores and atomic variables	2
	Total:	28

Bibliography:

1. Radu HOBINCU, Călin BÎRĂ, Object Oriented Programming, electronic course files, https://curs.upb.ro/
2. Stanley B. Lippman , Josee Lajoie, Barbara E. Moo, ” C++ Primer, 5th Edition”, Addison Wesley, ISBN: 9780321714114, 2012
3. https://en.cppreference.com/w/cpp - C++ Language documentation

LABORATORY

Crt. no.	Content	No. hours
1	Recap – using the debugger in pointers and structs exercises	2
2	Exercises with classes, objects, methods, fields and constructors	2
3	Exercises with operator overloading, destructors and references	2
4	Exercises with inheritance, polymorphism and (pure) virtual methods	2
5	Exercises with exceptions	2
6	Laboratory practical assignment simulation	2
7	Laboratory practical assignment	2
	Total:	14

Bibliography:
Laboratory support materials
- https://wiki.dcae.pub.ro/index.php/Programare_Orientat%C4%83_pe_Obiecte_(C%2B%2B)

11. Evaluation

Activity type	11.1 Evaluation criteria	11.2 Evaluation methods	11.3 Percentage of final grade
11.4 Course	Assimilation of the theoretical notions and C++ syntax throughout the semester.	Four homework assignments of 5 points each, of medium to high complexity, which are automatically evaluated and graded by the Virtual Programming Lab plugin on the Moodle platform.	10
	Assimilation of the theoretical notions as a whole.	Final exam, composed of 50 questions, taken on the Moodle platform.	40

11.5 Seminary/laboratory/project	The ability to use theoretical concepts to solve programming assignments.	Programming assignments during each laboratory.	20
	The capacity to abstract a real problem and devise a software solution in the C++ programming language.	Laboratory assignment, automatically evaluated and graded by the Virtual Programming Lab plugin on the Moodle platform, consisting in a simple requirement, similar to the ones practiced in the lab, with the specific requirement for it to be implemented correctly and completely for the points to be awarded.	30
11.6 Passing conditions			
• 50% of the total points • 50% of the total laboratory points			

12. Corroborate the content of the course with the expectations of representatives of employers and representative professional associations in the field of the program, as well as with the current state of knowledge in the scientific field approached and practices in higher education institutions in the European Higher Education Area (EHEA)

Object-oriented programming is one of the most important steps taken in the evolution of programming languages towards a stronger abstraction in the implementation of programs. In this context, the Object-Oriented Programming course is the second one in the line of programming disciplines, started with Computer Programming and finishing with Data Structures and Algorithms, and represents the basis for understanding and using the newer approaches in programming: component-based and service-based. The course curriculum responds concretely to these current development and evolution requirements, subscribed to the European economy of services in the field of Computers and Information Technology (CTI). In this way, the graduates are provided with adequate competences with the needs of the current qualifications and a modern, qualitative and competitive scientific and technical training, which will allow them to be employed quickly after graduation, being perfectly framed in the policy of the Politehnica University of Bucharest, both from the point of view of the content and structure, as well as from the point of view of the skills and international openness offered to students.

Date	Course lecturer	Instructor(s) for practical activities
25.09.2025	Conf. Dr. Radu Hobincu	As. Drd. Ing. Alexandra ENESCU

Date of department approval	Head of department

Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti
Facultatea de Electronică, Telecomunicaţii şi
Tehnologia Informaţiei

Prof. Dr. Ing. Claudius DAN

Date of approval in the Faculty Council

26.09.2025

Dean

Prof. Dr. Mihnea Udrea