SYLLABUS

1. Data about the program of study

1.1	Institution	Technical University of Cluj-Napoca
1.2	Faculty	Faculty of Automotive, Mechatronics and Mechanical Engineering
1.3	Department	Department of Mechatronics and Machine Dynamics
1.4	Field of study	Mechatronics and Robotics
1.5	Cycle of study	Bachelor of Science
1.6	Program of study/Qualification	Mechatronics
1.7	Form of education	Full time
1.8	Subject code	4.00

2. Data about the subject

2.1	Subject name		Applied Informatics I	Subject code		4.00
2.2	Course responsible/lecturer		Sl.dr.ing. Radu Donca - Radu.Donca@mdm.utcluj.ro			
2.3	Teachers in charge of seminars		Sl.dr.ing. Radu Donca - Radu.Donca@mdm.utcluj.ro			
2.4 Year of study		2.5 Semester		2.6 Assessment		E
2.7 Subject category DID DOB	Formative category					DF
	Optionality					DOB

3. Estimated total time

3.1 Number of hours per week	4	of which	3.2 Course	2	3.3 Seminar		3.3 Laboratory	2	3.3 Project	
3.4 Total hours in the curriculum	56	of which	3.5 Course	28	3.6 Seminar		3.6 Laboratory	28	3.6 Project	

3.7 Individual study:	
(a) Manual, lecture material and notes, bibliography	26
(b) Supplementary study in the library, online and in the field	5
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays	35
(d) Tutoring	0
(e) Exams and tests	3
(f) Other activities	0

3.8 Total hours of individual study (sum (3.7(a)…3.7(f)))	69
3.9 Total hours per semester (3.4+3.8)	125
3.10 Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	
4.2	Competence	

5. Requirements (where appropriate)

5.1	For the course	

5.2	For the applications	Attendance is compulsory, Personal Computer with specific software, specific teaching equipment

6. Specific competences

Professional competences	After completing the course, students will know: - Fundamental concepts of applied informatics using dedicated software. - Basic elements of the language and its syntax. - Alternative and repetitive control structures. - Function definition. - Algorithms based on recursion. - Optimization algorithms. - Division of complex problems using the Divide-and-Conquer method. After completing the course, students will be able to: - Develop small to medium-complexity programs in C++. - Identify various language elements. - Debug and improve various source codes available online. - Write programs tailored to specific requirements. - Solve coding problems specific to mechatronic systems. After completing the course, students will be capable of: - Developing various applications using the C/C++ language. - Utilizing different specific development environments. - Simulating various types of mechatronic systems (industrial robots, mobile platforms, humanoid robots, etc.) and designing the software necessary for their control.
Cross competences	**CT1:** Fulfilment of professional tasks with precise identification of the objectives to be achieved, available resources, completion conditions, stages of work, working time, and corresponding deadlines. **CT2:** Responsible execution of tasks within a multidisciplinary team, assuming roles at different hierarchical levels. **CT3:** Identification of the need for continuous professional development and the efficient use of informational resources and tools for assisted professional communication and training (e.g., internet portals, specialized software applications, databases, online courses, etc.), both in Romanian and in an internationally used language.

7. Expected learning outcomes

Knowledge	- The student identifies and describes the operating and design principles of automatic control systems. - The student identifies and describes mechatronic systems and subsystems, their design principles and methods, as well as the specific techniques, tools, and modern practices used in the design of mechatronic systems and subsystems. - The student identifies and describes specific computational techniques used in the design of automated systems and subsystems.
Skills	- The student selects and designs an automatic control system. - The student selects and applies methods for evaluating mechatronic system designs. - The student selects and applies computational techniques specific to mechatronic systems.

Responsibility and autonomy	- The student decides on the type of automatic control system to be implemented in specific applications and is responsible for its proper operation. - The student develops schematics, structural and functional diagrams, graphical representations, and specific technical documents. - The student decides on the choice of algorithms to be used and their implementation using specific computational techniques.

8. Discipline objectives (as results from the key competences gained)

8.1	General objective	The design and implementation of computing applications specific to mechatronics.
8.2	Specific objectives	Knoledge and ability to use C/C++, using relevant Integrated Development Environments

9. Contents

9.1. Lecture (syllabus)	Number of hours	Teaching methods	Notes
1. Presentation of the course, course bibliography, access to literature. The concept of an algorithm. Logical diagrams.	2	Video projector, whiteboard, open discussion, debate, internet resources	
2. Principles of structured programming. Logical diagrams. Simple variables.	2		
3. Basic elements of the C/C++ language. Input-output operations. Displaying data on the screen. Reading information from the keyboard. Linear control structures. Numeric data types. Arithmetic operations.	2		
4. Alternative structures. If-Else and Switch-Case statements. Boolean type. Logical operations.	2		
5. Repetitive structure. For loop. Data arrays. Indexing. Operations with arrays.	2		
6. Repetitive structure. While and Do-While loops. Sorting algorithms. Matrices.	2		
7. Characters. ASCII code. Operations with characters. Character strings. Specific applications.	2		
8. Subroutines. Modular problem-solving. Local variables. Global variables.	2		
9. Functions. Returning results. Parameters. Passing by value or by reference.	2		
10. Recursion. Mathematically recursive functions. Divide-and-Conquer. Towers of Hanoi.	2		

11. Files. Access. Reading data from a file. Saving results to disk.	2		
12. Graphics. Initializing graphics driver, pixels, lines, circles. Simple animations.	2		
13. Object-oriented programming. Classes/objects. Methods. Constructors.	2		
14. Object-oriented programming. Encapsulation. Inheritance. Polymorphism.	2		

Bibliography

1. Stanley Lippman - C++ Primer (5th Edition) - Addison-Wesley Professional, 2012
2. Bjarne Stroustrup - Programming: Principles and Practice Using C++ (2nd Edition) - Addison-Wesley Professional; 2nd edition 2014
3. Scott Meyers - Effective C++: 55 Specific Ways to Improve Your Programs and Designs 3rd Edition - Addison-Wesley Professional; 3rd edition, 2005
4. Andrei Alexandrescu - Modern C++ Design: Generic Programming and Design Patterns Applied 1st Edition, 2001
5. Thomas H. Cormen - Introducere in Algoritmi – Byblos, 2004
6. Gabriel Braharu - Limbajul C++ Pentru Incepatori (Volumul I + Volumul II), 2020
7. w3schools.com – C++ Tutorial - https://www.w3schools.com/cpp/
8. Robert C. Seacord, Effective C : an introduction to professional C programming, 2020, ISBN: 978-1-7185-0104-1

9.2. Laboratory	Number of hours	Teaching methods	Notes
1. Introduction – workplace safety rules, presentation of the working platform. Familiarization with the development environment.	2	Video projector, whiteboard, open discussion, debate, internet resources, lab applications	
2. Solving simple algorithms using logical diagrams and pseudocode descriptions. Variable tracking tables.	2		
3. Familiarization with the language. Data reading. Mathematical processing. Differences between numeric types. Displaying results.	2		
4. Problems with decision-making factors. Using alternative structures. Problems involving logical operations.	2		
5. Problems with fixed-step loops. Grouping data into arrays. Arithmetic operations (sum/product/average) on arrays. Examples of indexing problems.	2		
6. Loops with initial testing. Loops with final testing. Solving typical problems. Two-dimensional arrays.	2		
7. Processing character-type data. Modifications to a given text. Typical string problems. Text-to-numeric and numeric-to-text conversions.	2		

8. Exercises on defining custom subroutines. Decomposing a complex problem into simple modules. Variable visibility testing.	2		
9. Implementing mathematical functions. Testing results. Parameter passing by value and by reference.	2		
10. Solving mathematical functions defined recursively. GCD/LCM. Fibonacci sequence solved recursively. Typical recursion problems. Recursive algorithms.	2		
11. Text files. Typical problems. Modifying problems for reading data from files and writing results to files.	2		
12. Graphics problems. Activities with colors. Activities with drawing complex geometric shapes. Simple animations.	2		
13. Creating a class. Object instantiation. Examples, typical applications.	2		
14. Development of a class. Examples of derived classes. Applications of inheritance and polymorphism.	2		

Bibliography

1. Stanley Lippman - C++ Primer (5th Edition) - Addison-Wesley Professional, 2012
2. Bjarne Stroustrup - Programming: Principles and Practice Using C++ (2nd Edition) - Addison-Wesley Professional; 2nd edition 2014
3. Scott Meyers - Effective C++: 55 Specific Ways to Improve Your Programs and Designs 3rd Edition - Addison-Wesley Professional; 3rd edition, 2005
4. Andrei Alexandrescu - Modern C++ Design: Generic Programming and Design Patterns Applied 1st Edition, 2001
5. Thomas H. Cormen - Introducere in Algoritmi – Byblos, 2004
6. Gabriel Braharu - Limbajul C++ Pentru Incepatori (Volumul I + Volumul II), 2020
7. w3schools.com – C++ Tutorial - https://www.w3schools.com/cpp/
8. Robert C. Seacord, Effective C : an introduction to professional C programming, 2020, ISBN: 978-1-7185-0104-1

10. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

The content is updated in accordance with developments in the field and the needs of employers in the industry.

11. Evaluation

Activity type	10.1 Assessment criteria	10.2 Assessment methods	10.3 Weight in the final grade
11.4 Course	The final consists of a multiple-choice section (1/2 hour), summary topic section (1/2 hour) and	Written exam	50%

	application problems section (1 hour)		
11.5 Laboratory	Create clean and robust documentation, development of software solutions	Homework and lab work evaluation	50%
11.6 Minimum standard of performance			
Solving problems specific to the Mechatronics field based on the use of appropriate computational methods.			

Date of filling in:	Responsible	Title Surname Name	Signature
11.05.2026	Courses	Sl.dr.ing. Radu Donca - Radu.Donca@mdm.utcluj.ro	
	Applications	Sl.dr.ing. Radu Donca - Radu.Donca@mdm.utcluj.ro	

Approval from the Head of the Department coordinating the course, only if this is different from the department organizing the study program.	Head of Department Prof.Dr.Ing. Mircea Bara

Date of approval in the Department of Mechatronics and Machine Dynamics 28.05.2026	Head of Department Prof.dr.eng. Mircea Bara
Date of approval in the Faculty of Automotive, Mechatronics and Mechanical Engineering 16.09.2026	Dean Prof.dr.eng. Nicolae Filip