SYLLABUS

1. Program Information

1.1 Higher education institution	Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Department of Automation
1.4 Field of study	Automation, Applied Informatics and Intelligent Systems
1.5 Cycle of studies	Bachelor
1.6 Study Programme/Qualification	Intelligent Automation Systems (dual, in English language)
1.7 Form of education	IF – full-time education
1.8 Course code	1.00

2. Course information

2.1 Course title					Computer Programming and Algorithm Design	
2.2 Course lecturer					*Prof. dr. ing. Honoriu Vălean – honoriu.valean@aut.utcluj.ro*	
2.3 Seminar / Laboratory / Project Lecturer					*Prof. dr. ing. Honoriu Vălean – honoriu.valean@aut.utcluj.ro*	
2.4 Year of study	1	2.5 Semester	1	2.6 Type of assessment		E
2.7 Course status	Formative category *(DF, DS, DC)*					DF
	Optionality *(DOB, DOP, DFac)*					DOB

3.Total estimated time

3.1 Number of hours per week	4	of which:	HEI	Lecture	2	Seminar	0	Laboratory	2	Project	0
			CO		0		0		0		0
3.2 Number of hours per semester	56	of which:	HEI	Lecture	28	Seminar	0	Laboratory	28	Project	0
			CO		0		0		0		0

3.3 Distribution of time allocation (hours per semester) for:	HEI	CO
(a) Study based on textbook, course support, bibliography, and notes	20	0
(b) Additional documentation in library, specialized electronic platforms, and fieldwork	20	0
(c) Preparation of seminars/laboratories, assignments, papers, portfolios and essays	26	0
(d) Tutoring	0	0
(e) Examinations	3	0
(f) Other activities:	0	0
3.4 Total individual study hours (sum (3.3(a)... 3.3(f)))	69	0
3.5 Total hours per semester (3.2+3.4)	125	0
3.6 Number of credits per semester	5	0

(HEI = Higher Education Institution, CO = Company)

4. Prerequisites (where applicable)

4.1 Curriculum Prerequisites	• Basic programming skills in C/C++ language.
4.2 Competency Prerequisites	• Logical reasoning. • Problem-solving abilities. • Analytical thinking.

5. Conditions (where applicable)

5.1. Course Organization Conditions	• Blackboard/Smartboard, Projector.
5.2. Seminar / Laboratory / Project organization conditions	• Computers equipped with integrated development environments (IDEs) capable of compiling and executing C programs. • Laboratory attendance is mandatory.

6. Specific Competencies Acquired

Professional Competencies	• PC06 - Define technical requirements • PC07 - Demonstrate disciplinary expertise • PC17 - Operate open source software • PC23 - Synthesise information • PC24 - Think abstractly • PC26 - Use information technology tools
Transversal Competencies	• TC02 - Think analitically • TC03 - Demonstrate responsability

7. Learning outcomes

Knowledge:	• Understand the fundamental concepts of structured programming using the C language. • Describe and analyze the principles of algorithms and data structures. • Explain different algorithm design techniques such as recursion, divide and conquer, greedy methods, and dynamic programming. • Identify and evaluate the efficiency and computational complexity of algorithms.
Skills:	• Write, debug, and test C programs using appropriate programming constructs. • Design and implement algorithms for problem-solving using data structures such as lists, stacks, queues, trees, and graphs. • Apply modular and structured programming techniques for software development. • Evaluate and compare algorithm performance through experimentation and analysis.
Responsibility and autonomy:	• Work independently and collaboratively to design and implement programming solutions. • Demonstrate responsibility in testing, debugging, and documenting code. • Manage time and resources effectively during laboratory and project activities. • Show initiative in exploring alternative solutions and optimizing algorithm performance.

8. Course Objectives

8.1 General objective of the course	• Provide students a solid foundation in computer programming and algorithm design, enabling them to develop efficient, structured, and well-documented software solutions using the C programming language.
8.2 Specific objectives	• Understand and apply the fundamental concepts of structured and modular programming. • Design and implement algorithms for problem-solving using appropriate data structures such as lists, trees, graphs or hash tables. • Develop programs in C that demonstrate efficient use of functions, pointers, and file handling. • Analyze and compare algorithm performance and computational efficiency. • Apply systematic testing, debugging, and documentation practices in program development.

9. Contents

9.1 Lectures	No. of hours	Teaching methods	Obs.
Introduction and brief history of computing. Overview of hardware components.	2	Presentations; Demonstrations;	

Problem-solving steps and methodologies. Introduction to the C programming language: variables and expressions.	2	Collaborative Learning; Case Studies and Code Analysis; Interactive Discussions and Q&A Sessions;	
Arrays and statements.	2		
Functions and header files.	2		
Pointers and pointer operations.	2		
Structures, unions, and enumerations.	2		
Bit fields. Character and string functions. Standard library functions in C.	2		
Working with files and data streams.	2		
Recursion and functions with variable arguments.	2		
Modular programming principles.	2		
Module testing using the Google Test framework.	2		
The C preprocessor: directives and macros. Introduction to assembly programming in C.	2		
Basic concepts of concurrent programming and embedded systems.	2		
Syntactic and semantic error handling in C.	2		

Bibliography:

[1] T. Bailey, “An Introduction to the C Programming Language and Software Design.” Available [Online]: https://www-personal.acfr.usyd.edu.au/tbailey/ctext/ctext.pdf.

[2] P. Deitel and H. Deitel, C: How to Program, 8th ed., Global ed. Harlow, England: Pearson Education, 2016. Available [Online]: https://faculty.ksu.edu.sa/sites/default/files/c_how_to_program_with_an_introduction_to_c_global_edition_8th_edition.pdf.

9.2 Seminar / laboratory / project	Hours HEI	Hours CO	Teaching methods	Obs.
Algorithms: definitions, fundamentals, and performance analysis.	2	0	Hands-On Practice; Guided Programming Exercises; Debugging and Code Analysis Activities;	
Lists: types and representations, including singly and doubly linked dynamic lists; traversing and managing lists.	2	0		
List operations: insertion, deletion, and updating of elements.	2	0		
Stacks and queues: principles and specific operations.	2	0		
Fundamental sorting algorithms: bubble sort, insertion sort, selection sort, merge sort, quicksort, counting sort, and radix sort.	2	0		
Sets and set operations.	2	0		
Trees: types and representations, including binary trees, balanced trees, and AVL trees. Creation and in-memory representation of trees. Tree traversal techniques. Tree operations: insertion, deletion, and updating.	2	0		
Search algorithms in trees. Applications of trees in linguistics (2-3 and 2-3-4 trees) and in coding (Huffman coding).	2	0		
Graphs: types and representations, graph traversal techniques, and graph operations (insertion, deletion, updating).	2	0		
Problem-solving using graphs: shortest path algorithms (Dijkstra’s and Floyd’s algorithms) and minimum spanning tree algorithms (Kruskal’s and Prim’s).	2	0		
Hash tables: representation and applications.	2	0		

General methods for algorithm development: recursive algorithms, backtracking, divide and conquer, greedy method, and branch-and-bound technique.	2	0		
Algorithms: definitions, fundamentals, and performance analysis.	2	0		
Lists: types and representations, including singly and doubly linked dynamic lists; traversing and managing lists.	2	0		
Bibliography: [1] T. Bailey, "An Introduction to the C Programming Language and Software Design." Available: https://www-personal.acfr.usyd.edu.au/tbailey/ctext/ctext.pdf. [2] P. Deitel and H. Deitel, C: How to Program, 8th ed., Global ed. Harlow, England: Pearson Education, 2016. Available [Online]: https://faculty.ksu.edu.sa/sites/default/files/c_how_to_program_with_an_introduction_to_c_global_edition_8th_edition.pdf.				

10. Correlation of course content with the expectations of the epistemic community representatives, professional associations, and major employers in the field related to the program

The course content is aligned with the curricula of other technical universities and provides the essential knowledge and practical skills required by industry and employers in the areas of programming, software development, and algorithm design.

11. Evaluation

Activity Type	Evaluation criteria	Evaluation methods	Weight in final grade
11.1 Lecture	Understanding of theoretical concepts and problem-solving ability.	Written Exam	60%
11.2 Seminar/ Laboratory/Project	Application of programming and algorithm design concepts. Accuracy and clarity in presenting results.	Laboratory Test	40%
11.3 Minimum Performance Standard Laboratory test grade ≥ 5; Exam grade ≥ 5;			

Date of completion:	Lecturers	Title First Name Last Name	Signature
15.09.2025	Course	*Prof. dr. ing. Honoriu Vălean*	
	Applications	*Prof. dr. ing. Honoriu Vălean*	

Date of approval by the Department of Automation Council 24.11.2025 Date of approval by the Faculty of Automation and Computer Science Council 28.11.2025	Director of the Department of Automation Prof.dr.ing. Honoriu VĂLEAN Dean Prof.dr.ing. Vlad MUREȘAN