



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	Computer Programming and Programming Languages						
2.2 Course Activities Holder	Mihai L. MOCANU, Ph.D., Professor						
2.3. Practical Activities Holder	Cristian M. MIHAESCU, Ph.D., Professor						
2.4. Study Year	I	2.5. Semester	I	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	2	3.3. sem./lab./pr.	3
3.4. Total hours of curriculum	70	in which: 3.5 course	28	3.6. sem./lab./pr.	42
3.7. Time distribution - hours/week	55/14				
Study after manual, course support, bibliography and notes	21				
Additional documentation in library, on specialized electronic platforms and field	7				
Training seminars / labs, homework, portfolios and essays	14				
Tutorial	3				
Examinations	6				
Other activities.....	4				
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	Course is geared for people who have no previous computer programming experience. But, in order to complete this course successfully, students should be prepared to spend about five hours per week reading and solving problems in front of a computer, in addition to regular class attendance and active in-class participation (nobody learned how to program just from lectures, continuous problem solving/ experimentation using the computer is required).
4.2. of competence	None

5. CONDITIONS (where appropriate)

5.1. of the course	Lectures are held in university lecture halls equipped with the necessary audio-visual equipment. The textbook, written by the course instructor, is updated and published annually. All course materials—including documentation, presentations, samples from past examinations, quizzes, and seminar problems—are available on the course website.
5.2. of seminar/ laboratory/project	Seminars are held in university classroom settings in an interactive manner, using whiteboard support. They include a brief tutorial followed by illustrative problem-solving sessions. Emphasis is placed on understanding algorithms, problem-solving principles, and procedural programming concepts (in C). Homework will be assigned for each topic discussed, with a standard submission deadline of two weeks.

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

Knowledge	Student/Graduate: 1. Identifies and describes basic concepts, principles and methods in mathematics, physics, chemistry, technical drawing, economics and computer science. 2. Describes, identifies, and summarizes elementary concepts and methods regarding programming languages, programming environments, programming techniques.
Skills (Abilities)	Student/Graduate: 1. Operates with basic concepts, principles and methods from mathematics, physics, chemistry, technical drawing, economics and computer science. 2. Chooses and explains own concepts specific to algorithm design, object-oriented programming, logical and functional programming. The student/graduate specifies requirements, analyzes, designs, develops and tests software systems in general-purpose programming languages (i.e. C)
Responsibility and autonomy	Student/Graduate: 1. Applies the values of ethics and deontology of the engineering profession, practices logical reasoning, evaluation and self-evaluation in decision-making. 2. Has an honorable, responsible, ethical behavior, in the spirit of the law to ensure the reputation of the profession.

7. CONTENT

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction	On site (face to face)	Lectures will be delivered through interactive presentations using computers and specialized software, such as PowerPoint and an Integrated Programming Environment (IDE). Quizzes and discussions are scheduled to take place at the end of each lecture. Practical experience with the C programming language will be emphasized during all activities, and updates and	2
2. Hardware and Software	On site (face to face)		2
3. Programming Languages and Paradigms	On site (face to face)		2
4. Algorithm Design	On site (face to face)		2
5. Expressing and Implementing Algorithms	On site (face to face)		2
6. Algorithm Analysis, Theoretical and Practical Validation	On site (face to face)		2
7. Data Structures	On site (face to face)		2
8. C: Program Structure and Functions	On site (face to face)		2
9. C: Preprocessing. Basic I/O Operations	On site (face to face)		2
10. C: Data and Control Structures	On site (face to face)		2

11. C: Arrays and Pointers	On site (face to face)	improvements will be noted..	2
12. C: File I/O	On site (face to face)		2
13. C: Building Projects	On site (face to face)		2
14. C: The Standard Library	On site (face to face)		2

Bibliography (A-Core text, B-Secondary text, C-peripheral reading):

[A1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022. *(The updated standard; completely overhauls pseudocode, adds edge-based flows, and integrates modern computational problems).*

[A2] J. Eriksson, *Mastering Algorithms: A Practical Guide to Data Structures and Applications*. New York, NY, USA: Independently Published, 2026. *(A definitive modern handbook focusing on space-time complexities and practical system engineering).*

[A3] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. (Deluxe Syllabus Update). Boston, MA, USA: Addison-Wesley, 2016 (Reissued 2021). *(Focuses heavily on the structural implementation of modular data structures).*

[A4] K. N. King, *C Programming: A Modern Approach*, 2nd ed. (Modernized Printing). New York, NY, USA: W. W. Norton & Company, 2018. *(Widely adapted across global academic institutions for modern C foundations).*

[A5] M.Mocanu, *C: A Programming Guide*, Ed. Sitech, 2001 (in Romanian)

[B1] S. S. Skiena, *The Algorithm Design Manual*, 3rd ed. Cham, Switzerland: Springer, 2020. *(Features a highly practical breakdown of algorithm engineering alongside a catalog of real-world design resources).*

[B2] A. Forouzan, *Computer Science: A Structured Programming Approach Using C*, 4th ed. Boston, MA, USA: Cengage Learning, 2023. *(Perfect introduction to university teaching procedural code maps from scratch).*

[B3] D.E. Knuth, *The Art of Computer Programming - vol.1: Fundamental Algorithms*, 3rd ed., Addison Wesley Longman, 1997 (also translated in Romanian, Ed.Teora, 1999)

[B4] E.Horowitz, S.Sahni. *Fundamentals of Computer Algorithms*, Computer Science Press, 1984

[B5] E.Horowitz, S.Sahni. *Fundamentals of Data Structures*, Computer Science Press, 1986

[C1] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed. (Digital & Print Pearson Master Reissue). Hoboken, NJ, USA: Pearson, 2021. *(Though structurally written in 1988, this definitive reissue updates layout and syntax delivery for modern student access).*

[C2] J. T. McKinney, *Procedural Programming at Scale: Exploring Advanced Code Modularity and Logic*. Chicago, IL, USA: Open Academic Press, 2024. *(An specialized analysis of managing dynamic memory pools, stack optimization, and procedural structure scalability).*

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
Seminary:		During seminar and laboratory hours, students develop applications inspired by various engineering fields. Students are expected to study and experiment with algorithms, data structures, and other program components. They must learn to be systematic in their development process, to create functioning programs, while becoming familiar with at least one programming language (PL) and IDE (Integrated Developm. Environment).	
1. Mathematical Foundations of Computer Science. Binary/ decimal code conversions	On site (face to face)		2
2. A (Pascal-like) pseudocode language. Basic control structures. Data description	On site (face to face)		2
3. Programs using decision structures, iterations and recursions	On site (face to face)		2
4. Programs with linear static and dynamic data structures (arrays and lists)	On site (face to face)		2
5. Sorting and searching. Programs with tree data structures	On site (face to face)		2
6. Programming strategies (1): Greedy, Backtracking	On site (face to face)		2
7. Programming strategies (2): Divide&Conquer, Branch&Bound	On site (face to face)		2
Laboratory:			
1. Introductory activities	On site (face to face)		2
2. The C Compilation Model	On site (face to face)		2
3. The structure of C programs	On site (face to face)		2
4. Representation and Control Flow / Iteration and Recursion	On site (face to face)		2
5. Using Functions	On site (face to face)		2
6. Arrays and Strings. Other	On site (face to face)		2

Structured Types			
7. Pointers - Introduction	On site (face to face)		2
8. Pointers – Dynamic Memory Allocation	On site (face to face)		2
9. Dynamic Data Structures	On site (face to face)		2
10. Sorting and Searching	On site (face to face)		2
11. Files and Streams	On site (face to face)		2
12. Programming Practices	On site (face to face)		2
13. Development Environment and Testing Architecture	On site (face to face)		2
14. Final Evaluation	On site (face to face)		2
Bibliography: [B6] P. Deitel and H. Deitel, <i>C How to Program: With an Introduction to C++</i> , 9th Global ed. London, UK: Pearson, 2022. (Provides an intense, code-dense methodology geared toward university lab applications). Some of the references above are also used, plus internal documentations for IDEs			

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

Course and application contents are in accordance with similar items from other universities in the country and abroad. Gained competences are convergent with the requirements of major employers, being the result of periodic discussions with representatives of the local companies active in the IT field.

9. EVALUATION

Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
9.4. Course	Procedural programming concepts understanding; problem correct solving and use of a procedural language (C); ability to connect concepts as taught; gain of analysis and synthesis skills.	(Periodic) Quizz(es) Final Exam, Questions & Exercises, written + oral	20% 40%
9.5. Seminar/laboratory/project	Concepts understanding; problem correct solving; ability to connect concepts as taught; gain of analysis and synthesis skills.	Written + oral verifications	20%
	Procedural programming concepts use; problem correct solving and use of the C procedural language.	Homeworks/ regular debates (usually every other week) + Final practical evaluation	20%
9.6. Minimum standard of performance Basic knowledge in solving problems using an algorithmic language. Active participation during at least 12 (of 14) laboratory sessions and proof of implementation abilities (in C).			

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
1.10.2025		

Date of Approval in the Department	Signature of the Department Director,
6.10.2025	Ș.l.dr.ing. Nicolae ENESCU

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