SYLLABUS

1. Data about the program of study

1.1 Institution	The Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Bachelor of Science
1.6 Program of study / Qualification	Computer science / Engineer
1.7 Form of education	Full time

2. Data about the subject

2.1 Subject name	***Computer Programming***			Subject code	**5**
2.2 Course responsible / lecturer	Lect. dr. eng. Joldoş Marius - Marius.Joldos@cs.utcluj.ro Lect. dr. eng. Varga Robert - Robert.VARGA@cs.utcluj.ro				
2.3 Teachers in charge of seminars / Laboratory / project	Lect. dr. eng. Varga Robert - Robert.VARGA@cs.utcluj.ro Lect. dr. eng. Pocol Ciprian - Ciprian.Pocol@cs.utcluj.ro Assist. drd. eng. Chiciudean Vivian - Vivian.Chiciudean@cs.utcluj.ro				
2.4 Year of study	I	2.5 Semester	1	2.6 Type of assessment (E - exam, C - colloquium, V – verification)	E
2.7 Subject category	Formative category: DF – fundamental, DS – speciality, DC – complementary				DF
	Optionality: DOB – mandatory, DOP – optional (alternative), DFA – optional (free choice)				DOB

3. Estimated total time

3.1 Number of hours per week	5	of which:	Course	2	Seminars	1	Laboratory	2	Project	-
3.2 Number of hours per semester	70	of which:	Course	28	Seminars	14	Laboratory	28	Project	-
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										25
(b) Supplementary study in the library, online and in the field										20
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										25
(d) Tutoring										5
(e) Exams and tests										5
(f) Other activities:										-

3.4 Total hours of individual study (suma (3.3(a)...3.3(f)))	80
3.5 Total hours per semester (3.2+3.4)	150
3.6 Number of credit points	6

4. Pre-requisites (where appropriate)

4.1 Curriculum	N/A
4.2 Competence	N/A

5. Requirements (where appropriate)

5.1. For the course	N/A
5.2. For the applications	N/A

6. Specific competence

6.1 Professional competences	**C1 –** Operating with basic Mathematical, Engineering and Computer Science Concepts. • **C1.1 -** Recognizing and describing specific concepts to calculability, complexity, programming paradigms and modeling of computing and communication systems • **C1.2 -** Using specific theories and tools (algorithms, schemes, models, • protocols, etc.) for explaining the structure and the functioning of • hardware, software and communication systems • **C1.3 -** Building models for various components of computing systems • **C1.4 -** Formal evaluation of the functional and non-functional characteristics of computing systems • **C1.5 -** Providing theoretical background for the characteristics of the designed systems
6.2 Cross competences	N/A

7. Expected learning outcomes

7.1 Knowledge	• **Recognize** the binary representation of information and the specific memory footprints of simple data types to understand how a digital system stores and interprets various forms of data. • **Comprehend** the theoretical stages of algorithmic design using pseudocode and block diagrams to translate human logic into the structured requirements of compiled languages. • **Internalize** the operational logic of sequential, conditional, and repetitive statements as the essential building blocks for directing data flow and decision-making within a digital system. • **Master** the principles of code modularization, specifically the theoretical boundaries of functions, variable scope, and the use of header-based interfaces to create scalable and encapsulated software architectures. • **Discern** the relationship between abstract data and physical hardware by understanding the memory representation of arrays and strings alongside the power of pointers to directly manipulate system resources. • **Synthesize** the methods for grouping heterogeneous data through structures and the theoretical requirements for long-term data persistence via high-level file I/O.
7.2 Abilities and Skills	• **Analyze** computational problems to determine the logical steps and data types required to produce a valid digital solution. • **Design** modular software architectures by decomposing problems into functions and using header-based interfaces to organize the program structure. • **Interpret** existing source code to trace data flow, track memory addresses, and explain the execution path of a program. • **Implement** programs in pure C that combine basic control statements and pointers to perform direct memory and data manipulation. • **Diagnose** logical errors and memory-related defects, such as invalid pointer references or uninitialized variables, through manual code inspection. • **Correct** defective source code by resolving logical inconsistencies and syntax errors to ensure the program executes as intended. • **Develop** simple system applications that group related data using structures and manage information across sessions via file I/O.

7.3 Responsibility and Autonomy	• **Organize** personal work within the lab session to ensure the timely completion of exercises while managing individual progress effectively. • **Exercise** autonomy by independently utilizing technical documentation and compiler diagnostics to troubleshoot errors before seeking instructor intervention. • **Articulate** the logic and structure of one's own code through clear and concise verbal explanations during supervised laboratory evaluations. • **Assume** responsibility for the integrity of personal work by ensuring code is original, logically sound, and follows memory-safety best practices. • **Manage** a personal development environment, including the configuration of compilers, text editors, and the systematic organization of project files on an individual workstation. • **Refine** programming solutions based on instructor feedback, demonstrating a proactive approach to improving code quality and technical self-sufficiency.

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

8.1 General objective	To learn how to use a general-purpose, high-level programming language for writing programs
8.2 Specific objectives	• To understand a small-sized problem stated in a natural language, and to develop a solution as a computer program. • To understand code written by other programmers and to reason critically about them. • To design and implement computer programs in C using the structured/modular approach. • To learn a good programming style. • To determine the causes of programming errors and to correct them.

9. Contents

9.1 Lectures	Hours	Teaching methods	Notes
Programming Languages. Stages of Problem solving Using Computers. Algorithm – Definition, Properties. C features. Simple Data Types. Simple I/O	2	Lectures, demos and discussions	Uses a video-projector
Programming Style. Digital Representations. Variables and Expressions	2		
C Statements. C Preprocessing	2		
Functions (Structure, Invocation, Parameter passing, Functions as parameters, Variable scope). Functions for character processing	2		
Modular Programming. Debugging	2		
Pointers (I). Pointer variables. Pointer arithmetic. Pointers as arguments and return values	2		
Pointers (II). Pointers and Arrays. Memory management. Pointers to Pointers. Function Pointers	2		
Recursion	2		
C Character Strings. C library	2		
Structures, unions, enumerations. User-defined Types	2		
File Handling. High Level I/O.	2		
Advanced use of learned concepts	2		
Review	2		

Bibliography:

1. Paul and Harvey Deitel, C: How to program, Pearson Education, 6ed, 2010
2. K.N. King, C Programming: A modern Approach, W.W. Norton, 2008
3. Stephen Prata, C Primer Plus, Sams, 5ed, 2004
4. Brain W. Kernighan, Dennis M. Ritchie – The C Programming Language, Prentice Hall, Inc., 1988.

9.2 Applications - Seminars / Laboratory / Project	Hours	Teaching methods	Notes
S1. Algorithm Representations (Flowcharts, Pseudocode)	1	Tutoring, discussions, and in-class problem solving, Tutoring, discussions, and assisted program development.	PCs equipped with MinGW C development kit and Codeblocks IDE
S2. Operators, Expressions, Functions	1		
S3. Functions and Modular Programming	1		
S4. Pointers and Memory Management	1		
S5. Recursion. String Manipulation	1		
S6. Structures, Unions, Enumerations	1		
S7. Working with Files. Command line arguments	1		
L1. Pseudocode. Interactive Development Environments for C. Setting up and Using Codeblocks IDE	2		
L2. C data types. Simple IO in C	2		
L3. Operators and Expressions in C	2		
L4. Statements in C	2		
L5. Functions. Debugging C programs	2		
L6. Modular Programming	2		
L7. Pointers (I). Pointers and Arrays	2		
L8. Pointers (II) and memory management	2		
L9. Recursion	2		
L10. Character string manipulation	2		
L11. Structures, Unions, Enumerations	2		
L12. Recursion, High level I/O in C. Command line arguments	2		
L13. Review	2		
L14. **Laboratory test**	2		

Bibliography:

1. Moodle site for course available at: https://moodle.cs.utcluj.ro/ (laboratory session description are available on the site)

**Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

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

C1.4 - Formal evaluation of the functional and non-functional characteristics of computing systems

11. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	Written exam	Three in-class tests (T) + Final Written exam (W)	60% = 50% W + 10% T
Seminar	Homework and Seminar activity	Homework tasks	
Laboratory	Laboratory test	Evaluation of the test solutions	40%
Project	-	-	-

Minimum standard of performance: evaluation grade ≥ 5 Grade calculus: 40% laboratory + 60% exams and tests
Conditions for participating in the final exam: Laboratory ≥ 5
Conditions for promotion: final written exam grade ≥ 5 and final written exam problems grade ≥ 5

Date of filling in: 29.04.2026	Responsible	Title, First name Last name	Signature
	Course	Lect.dr. eng. Marius JOLDOȘ	
		Lect.dr. eng. Robert VARGA	
	Applications	Lect.dr. eng. Robert VARGA	
		Lect.dr. eng. Ciprian POCOL	
		Assist.drd. ing. Vivian CHICIUDEAN	

Date of approval in the department	Head of department, Prof.dr.eng. Rodica Potolea
Date of approval in the Faculty Council	Dean, Prof.dr.eng. Vlad Mureșan