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 1 Computer Programming and Programming Languages 1			
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	I	2.6. Evaluation type	E	2.7 Course regime	Ob
2.8 Course type		F	2.9 Course code	04.F.01.O.004		2.10 Tipul de notare	Nota

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

3.1 Number of hours per week	4	Out of which: 3.2 course	2.00	3.3 seminary/laboratory	2
3.4 Total hours in the curricula	56.00	Out of which: 3.5 course	28	3.6 seminary/laboratory	28
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.					42
Tutoring					2
Examinations					0
Other activities (if any):					0
3.7 Total hours of individual study	44.00				
3.8 Total hours per semester	100				
3.9 Number of ECTS credit points	4				

4. Prerequisites (if applicable) (where applicable)

4.1 Curriculum	Basic algebra.

4.2 Results of learning	Generic knowledge of computer handling.

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 (gcc), 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)*

This course is fundamental to the ETTI field and aims to familiarize students with a general purpose programming language (C). The discipline is an introductory one, which begins with general concepts of the imperative-procedural paradigm, continuing with the description of the syntax of the language. At the same time, the discipline aims to accustom the student to work in a manner focused on the implementation of functional solutions, the evaluation being done automatically, by a software platform configured through Input-Output tests. This represents the basis for data acquisition and processing as well as using more advanced computation systems.

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	C2. Common use of basic methods for signal data acquisition and processing (1/4). C3. Use of computational systems architecture, microcontrollers and computer programming knowledge, concepts and basic apparatus (3/4).
Transversal (General) Competences	Developing logical, structured cognitive abilities, and gaining the ability to break-down and analyze a given problem.

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 how to compile programs into applications, as well as how programs work at runtime.

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 a personal solution to solve the problem, as well as the ability to implement this solution in C language.
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	Software Development Life-Cycle - SDLC – Architecture and Design; Basic notions about the compiler, formal language and development environments; Hello world!	1
2	The variable and assignment operator – lvalue and rvalue	1
3	Numeric data types – int and double; Process streams; Formats for reading and writing data (printf & scanf); Arithmetic operators	3
4	Numerical bases – binary, decimal and hexadecimal; Storing negative integers - two's complement; Fixed-size integer data types; Bitwise operators; Storage of Floating Point Numbers – IEEE 754 Standard	3
5	Conditional statements: if-else, switch-case and the conditional operator; Logical operators and evaluating logical expressions	2
6	Cast operator; Loops - while, do-while, for. Sequence operator (,)	2
7	Uni- and multi-dimensional arrays; Character storage – ASCII and Unicode codes; Strings	4
8	Functions, local and global variables; Pass-by-value vs. Pass-by-reference; Memory segments; Program execution model; Recursive functions; Tail recurrence	4
9	Data aggregation: struct, enum and union	2

10	Pointers and pointer arithmetic; Dynamic memory allocation; Handling files	6
	Total:	28

Bibliography:

1. Radu HOBINCU, Călin BÎRĂ, Computer Programming, electronic course files, https://curs.upb.ro/
2. Greg Perry, Dean Miller, ”C Programming Absolute Beginner's Guide, 3rd Edition”, Que Publishing, ISBN: 0789751984, 2013
3. http://en.cppreference.com/w/c - C Language API documentation
4. https://gcc.gnu.org/onlinedocs/gcc/ - gcc compiler documentation

LABORATORY

Crt. no.	Content	No. hours
1	Linux – basic Bash commands	2
2	GNU Compiler Collection, GNU Make	2
3	Standard process streams	2
4	Numeric data types and operators	2
5	Conditional expressions	2
6	Loops	2
7	Arrays	2
8	Strings and the string manipulation library	2
9	Functions	2
10	Recursive functions; The GNU Debugger - gdb	2
11	Structures (struct, union and enum)	2
12	Pointers; The valgrind profiler	2
13	File manipulation	2
14	Laboratory practical assignment	2
	Total:	28

Bibliography:

Laboratory materials - https://wiki.dcae.pub.ro/index.php/Programarea_Calculatoarelor_(laborator)

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 the C language syntax throughout the semester.	Ten homework assignments of medium to high complexity, which are automatically evaluated and graded by the CodeRunner plugin on the Moodle platform.	10
	Assimilation of the theoretical notions and the C language syntax as a whole.	Final exam, composed of 50 questions, taken on the Moodle platform.	40
11.5 Seminary/laboratory/project	The degree in which the student has fulfilled the requirements for individual study.	Multiple choice test at the start of each laboratory.	5
	The ability to use theoretical concepts to solve programming assignments.	Programming assignments during each laboratory.	15
	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 CodeRunner 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			
1. Obtaining 50% of the total points. 2. Obtaining 50% of the total laboratory points (25/50).			

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)

Computer programming is the first step towards the wide domain of software engineering. Nowadays, when even circuit and systems design are done by using dedicated hardware description languages, it is unconceivable for a graduate from a technical university to be ignorant in the field of programming. This acute need for programming knowledge was repeatedly mentioned by representatives of the IT industry in relation with the academic staff. Therefore, our faculty has taken additional measure to improve this skill among our graduates.

Date Course lecturer Instructor(s) for practical activities

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

Conf. Dr. Radu Hobincu

As. Drd. Ing. Costin-Emanuel VASILE

As. Drd. Ing. Andrei-Alexandru ULMĂMEI

As. Drd. Ing. Alexandra ENESCU

As. Drd. Ing. Alexandru GUZU

Date of department approval

Head of department

Prof. Dr. Ing. Claudius DAN

Date of approval in the Faculty Council

Dean

Prof. Dr. Ing. Radu-Mihnea UDREA