SYLLABUS[1]

1. **Information about the Program**

1.1 Higher education institution	Politehnica University of Timişoara
1.2 Faculty[2] / Department[3]	Automation and Computers/ Computers
1.3 Chair	-
1.4 Domain of study	Computers and Information Technology
1.5 Study level	Bachelor
1.6 Study programme / Qualification	Computers / engineer

2. Information about the Course

2.1 Course			Computer Programming				
2.2 Lecturer			dr. Marius Minea				
2.3 Academic staff for seminars/labs			ing. Alexandru Iovanovici				
2.4 Study year	1	2.5 Semester	1	2.6 Assessment type	E	2.7 Course type	Mandatory

3. Total time estimated (hours/ semester of didactical activities)

3.1 Hours / week	5	of which:	3.2 lecture hours	3	3.3 seminar/lab hours	2
3.4 Total curriculum hours	118	of which:	3.5 lecture hours	42	3.6 seminar/lab hours	28
Time distribution						hours
Study using manuals, support materials, bibliography and notes						20
Supplementary documentation in library, speciality electronic platforms and on site						7
Supplementary preparation for seminars/labs, homeworks, reviews, portofolios and essays						14
Tutoring activities						14
Exams						3
Other						

3.7 Total - hours of individual study	111
3.8 Total - hours per semester	128
a. **Credits**	5

4. Prerequisites (if appropriate)

4.1 curriculum	• none
4.2 competencies	• basic mathematical and logical abilities

5. Conditions (if appropriate)

5.1 for lectures	• large classroom, laptop, projector, blackboard/whiteboard
5.2 for seminars/labs	• lab with 15-18 computers, C programming environment, blackboard/whiteboard

6. Specific competencies acquired

Professional competencies[4]	• Working with foundational concepts of the sciences, engineering, and computer science • Design of hardware, software and communication components • Solving problems using computer science and engineering tools

1 Formularul corespunde Fişei Disciplinei promovată prin OMECTS 5703/18.12.2011 (Anexa3);

2 Se înscrie numele facultăţii care gestionează programul de studiu căruia îi aparţine disciplina;

3 Se înscrie numele departamentului căruia i-a fost încredinţată susţinerea disciplinei şi de care aparţine titularul cursului;

4 Aspectul competenţelor profesionale va fi tratat cf. Metodologiei OMECTS 5703/18.12.2011. Se vor prelua competenţele care sunt precizate în Registrul Naţional al Calificărilor din Învăţământul Superior RNCIS (http://www.rncis.ro/portal/page?_pageid=117,70218&_dad=portal&_schema=PORTAL) pentru domeniul de studiu de la pct. 1.4 şi programul de studii de la pct. 1.6 din această fişă.

Transversal competencies	• Honorable, responsible, ethical and lawful behavior in solving the assigned problems • Showing action and initiative to update professional, economic, and organizational knowledge

7. Objectives of the course (issued from the list of the competencies acquired)

7.1 Aim	• Understand and use fundamental programming language concepts.
7.2 Specific objectives	• Develop basic algorithmic skills. Solve small-scale problems by writing programs in C. Appreciate clean coding style and program correctness. Understand basics of machine-level program representation.

8. Content

8.1 Lecture	Hours	Instruction methods
1. Functions as fundamental programming construct: parameters, function definition and function call, basic types, composing functions; computation vs. input/output, conditionals	2	lecture with slides, questions, explanations, live programming with examples
2. Recursion: recurrent sequences, fractals, expressions. Mechanism of a recursive function call. Efficiency. Tail recursion.	5	
3. Character-based I/O reading and writing characters; converting to a number;	2	
4. Imperative programming recursion vs. iteration; assignment; loops; character-based filters	3	
5. Internal representation representation of integers and reals; bitwise operators	3	
6. Arrays: vectors; matrices; strings	6	
7. Input/output input checking and error handling; formatted I/O	5	
8. Pointers: addresses of variables; pointer arithmetic; dynamic allocation	6	
9. Files: text files; binary files	4	
10. Structures structures and unions; defining new datatypes	3	
11. Modularization. Abstract datatypes preprocessor; C files and headers; interface and implementation	3	
References Allen B. Downey. *How to think like a computer scientist*. C version by Thomas Scheffler, 2010 Tim Bailey. *An Introduction to the C Programming Language and Software Design*, 2005 Brian Kernighan, Dennis Ritchie. *The C programming Language*, 2nd. ed., Prentice Hall, 1988		
8.2 Seminar/lab	Hours	Instruction methods
1. Programming environment. Compilation. Simple programs with conditionals	2	programming and problem solving, homework, explanations, discussion,
2. Recursion. Examples with sequences, series.	2	
3. Fractals. Tail recursion.	2	
4. Iterative vs. recursive processing – numeric examples.	2	
5. Iterative text processing. Filters.	2	
6. Bitwise operations. Simple image processing. Polynomial division. Checksums	2	
7. Arrays. Basic searching, sorting. String processing.	2	
8. Input/output: finding patterns in texts. Processing structured text.	2	
9. Pointers. String functions	2	
10. Pointers: dynamic allocation, resizing arrays, pointer arrays	2	
11. Files. Processing images, archives.	2	
12. Structures as compound values. Processing formatted files (csv)	2	
13. Implementing a simple library (e.g. lists)	2	
14. Recap and make-up lab.	2	
References Stephen Kochan, *Programming in C*, 3rd Edition, Sams Publishing, 2005 Peter van der Linden. Expert C Programming. Prentice Hall, 1994		

9. Correlation between the course content and the requirements of the specialists in the field and the expectations of the main employers

• Solid programming skills (including algorithmic skills, good error handling and avoiding security pitfalls) are fundamental for any software engineer position. C is often used itself (embedded automotive) or as a basis for learning other languages.

10. Assessment

Activity type	10.1 Assessment criteria	10.2 Assessment methods	10.3 Weight in final mark
10.4 Lecture	quiz: theory and code samples programming to solve problems	written	15%
		computer-based	35%
10.5 Seminar /labs	homeworks programming to solve problems	computer-based, written	15%
		computer-based	35%
10.6 Minimal performance standards (minimal specific knowledge required for passing the exam, the means to assess mastering the specific knowledge)			
• writing runnable programs that implement the core problem requirements, covering the main topics taught in class			

11. **International compatibility**

- INF-1-CP, Computer programming skills and concepts, Univ. Edinburgh. http://www.inf.ed.ac.uk/teaching/courses/cp1/
- 6.087. Practical Programming in C. MIT Open Courseware. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-087-practical-programming-in-c-january-iap-2010/
- Steven Summit. C Programming, Experimental College, Univ. of Washington. http://www.eskimo.com/~scs/cclass/cclass.html

Date	Signature of the course instructor	Signatures of the academic staff for seminars/labs
14.10.2013	dr. Marius Minea	ing. Alexandru Iovanovici

Date of approval in the Department

Signature of the Department Director

prof. dr. ing. Vladimir Creţu