RTU Course "Computer Studies (basic course)"

13223 Elektronikas pamatu katedra

General data

Code	RTR105
Course title	Computer Studies (basic course)
Course status in the programme	Compulsory/Courses of Limited Choice
Course level	Undergraduate Studies
Course type	Academic
Field of study	Electronics and Telecommunications
Responsible instructor	Viktors Zagorskis
Academic staff	Dmitrijs Puriševs Tatjana Solovjova Jurijs Ivanovs
Volume of the course: parts and credits points	1 part, 3.0 Credit Points, 4.5 ECTS credits
Language of instruction	LV, EN, RU, DE
Possibility of distance learning	Not planned
Abstract	The course aimed at the first year bachelor students of RTU Faculty of Electronics and Telecommunications provides insights into contemporary computers, modern programming languages and elementary computing algorithms to be applied in further studies and engineering work. The course outlines mathematical, engineering and philosophic principles of data acquisition, computing and representation systems, which provide the basis for development of professional expertise and practical competence working with professional computer systems based on open source operating environment Linux, UBUNTU and SuSe implementations in particular. The students get acquainted with classical programming languages C, C++ and contemporary programming language Python. Binary, hexadecimal and complex numbers computing systems are studied. The concepts of ‘bit’, ‘byte’ and ‘data type’ are comprehensively analysed. Practical programming tasks are related to simple tasks completed within the framework of the courses in physics, electronics and mechanical engineering, as well as to a range of themes in advanced mathematics such as simple function differentiation and square computing of functionally developed geometric figures. The tasks connected to complex numbers, bit operations and data sorting are completed depending on the specifics of the study program.
Goals and objectives of the course in terms of competences and skills	The goal of the course is to provide an overview of C, C++ and Python programming languages and free access technical tools for development of simple but professional software. An overview of modern, professionally efficient programming environment based on Linux operating systems is given. Students get practical experience developing algorithms able to solve numerically simple tasks in physics and mathematics with an aim to design simple C++ and Python scripts and develop procedural programming skills.
Structure and tasks of independent studies	During the semester the students take part in the following planned activities getting prepared for them individually. I. Lectures. 12 lectures starting from week 1, one per week. II. Home works 12 home works on the themes covered at the lectures. III. Labs.12 weekly labs, starting from week 3. IV. Midterm test. Week 9. Written test on C, C++ related issues VI. Final test. Week 16. Written test on Python related issues. Algorithm problems. VII. Exam. Oral practical problem solving exam. The student’s skills and competences in completing course related tasks are demonstrated.

Recommended literature	[1] Stroustrup, B. The Programming Language C++. Addison-Wesley, 2009. [2] The C++ Resources Network. Available: http://www.cplusplus.com/ [3] Elkner, J., Downey, A.B., Meyers, C. How to Think Like a Computer Scientist, Learning with Python. 2nd ed. 2008. Available: http://www.openbookproject.net/thinkcs/python/english2e/ [4] Elkner, J., Downey, A.B., Meyers, C., McCane, B., Hewson, I., Meek, N. Practical Programming in Python. 2009. Available: http://www.cs.otago.ac.nz/staffpriv/mccane/Downloads/PracticalProgramming.pdf [5] Knuth, D. All questions answered. Notices of the American Mathematical Society, vol. 49(3), 2002, p.318–324. [6] Herrmann, D. C++ für Naturwissenschaftler: beispielorientierte Einführung. Addison-Wesley, 2001. [7] Shader, M., Kuhlins, S. Programmieren in C++: Einführung in den Sprachstandard. Springer, 1998. [8] Dausmann, M., Broeckl, U., Goll, J. C als erste programmiersprache.Teubner, 2005. [9] Zagorskis, V. Datormācība-pamatkurss. Laboratorijas darbu apraksti. Rīga: RTU, ETF, SC, 2010. [10] Zagorskis, V. Datormācības lasījumi. I. daļa. Rīga: RTU, ETF,SC, 2009 [11] Course RTR105 class notes. Available: http://213.175.92.39
Course prerequisites	High school education level knowledge of subjects in mathematics and physics.

Course outline

Theme	Hours
Human and Computer. Similarities and differences. Interoperability philosophy. Information. Information unit. Bit. Byte	1
Operational systems (OS). Why OS systems are needed? Concept. History. How OSes are developed?	1
UNIX. Multiuser and multiprocess system. Time divided processes. Linux.	1
OS kernel. Shell. Physical devices. File systems.	1
Users in Linux. Terminal application. Shells. Shell commands. Variables, directories, files. Rights.	2
Data-In. Data-Processing. Data-Out. Programming in shell. Shell scripts.	1
C, C++ languages. Principles of languages. Syntax. Building blocks. Operators. Variables. Statements.	1
Main function in C, C++. How to compile C, C++ program? Pre-processing directives.	1
How to execute a binary file? Data evaluation. Data formatting. Standard output. Error output.	1
C, C++ libraries. Data type - char. Declaration and definition of variables.	1
Circle of numbers. Natural numbers and integers. Characters. ASCII. Arithmetical operations. Priorities.	1
Data exchange. Algorithms. Alg I. Dec to Bin conversion.	1
Different Integers. Short. Int. long data types. Library file limits.	1
Floating point numbers. Float, double, long double data types. Data arrays.	3
User defined C, C++ functions. Prototypes. Program block.	1
Looping with FOR and WHILE. Loop variable. Loop body.	1
Alg II. Factorial. Alg III. Bin to Dec conversion.	1
Some useful functions from STD - standard C library.	1
Data type - BOOLEAN. Program branching. Logical operations. Bitwise operations.	1
Data arrays. Pointers. Pointers arithmetic. References. User defined data types: structures, etc..	2
How to format data and write to file? How to read the file? Text and binary files.	1
Python programming language elements. Arithmetical operations. Data-In. Data-Out. Data types.	1
Data types in Python. Integers, floating point numbers, complex numbers.	2
Advanced data types: list, dictionary. Looping in Python. How to read data lines from file?	1
Alg. iV. Factorial. Alg. V. Number series. Recurrence.	2
Python standard libraries (sys, os, math, cmath, numpy, etc.). User functions. Lambda function.	2
Alg. VI. Finding of math-function root.	2
Branching in Python. IF. IFELSE. ELSE. Logical operations. Bitwise operations.	1
Alg. VII. Data sorting. Bubble algorithm.	2
Alg. VIII. Function differentiation.	2
Alg IX. Finding of function integral.	2
Scientific libraries for Python. SciPy. NumPy. Matplotlib. Independent tool GNUPLOT.	2
User programs for scientific usage (Octave, Maxima, etc.).	2
Elementary descriptive statistics.	2

Learning outcomes and assessment

Learning outcomes	Assessment methods
Skills in using professional UNIX like operational systems in terminal environment. Students have practical experience to manipulate directories, files, work with text editors, invoke scripts and programs	Labs, scored. Tests. Exam.
Ability to write simple C, C++ programs based on looping, branching and conditioning procedures. Knowledge of C, C++ program compiling, syntax error finding, executing, terminal output redirection	Labs, scored. Tests. Exam.
Skills in creating simple programs in Python based on some fundamental algorithms like function root finding, definitive integral value calculation, function differentiation, simple bubble sorting.	Labs, scored. Tests. Exam.
Students are able to operate GNUPLOT as math-functions 2D visualisation and evaluation tool.	Labs, scored. Tests. Exam.

Study subject structure

Part	CP	ECTS	Hours per Week			Tests		
			Lectures	Practical	Lab.	Test	Exam	Work
1.	3.0	4.5	1.5	0.0	1.5		*	