RTU Course "Algorithms and Methods of Programming"

12308 null

General data

Code	DIP321
Course title	Algorithms and Methods of Programming
Course status in the programme	Compulsory/Courses of Limited Choice
Course level	Undergraduate Studies
Course type	Academic
Field of study	Computer Science
Responsible instructor	Jurijs Lavendels
Academic staff	Gints Jēkabsons
Volume of the course: parts and credits points	1 part, 2.0 Credit Points, 3.0 ECTS credits
Language of instruction	LV, EN
Annotation	The aim of the course is to develop algorithmzation skills, practical software development.Topics covered by the course include: Definition of algorithm. Parts of algorithm theory. Algorithms and software. Communication and synchronization between running algorithms.
Goals and objectives of the course in terms of competences and skills	The results are achived providing: academic knowledge on parts of algorithm theory, properties and development, formal language theory, generative grammar,lexical analysis,implementation of recursion in development of a number software.
Structure and tasks of independent studies	Individual work consists of: - learning of methodological materials and literature; - practical development of algorithms.
Recommended literature	D. Gries. . Compiler construction for digital computers Wiley, 1992, 545 p. R. Mak. . Writing Compilers and Interpreters: An Applied Approach. Wiley, 2013, 520 p. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-035-computer-language-engineering-spring- -2010/lecture-notes/ MIT lecture notes, 2005 V. Aho, D.Ullman. The thory of parsing, translation and compiling Volume I Parsingvolume 2 Translation and Compiling Englewod Cliffs, Prentice-Hall, 1972, 1099 p L. Bauer, .L. DeRemer,M. Griffiths,U. Hill,.J. Hornig, H. Koster, M. McKeeman, C. Poole. Compiler Construction: An Advanced Course Berlin, Springer-verlag, 2014, 608 p. .
Course prerequisites	Programming languages, operating systems

Course outline

Theme	Hours
Definition of algorithm.	8
Formal language theory; generative grammar;	6
Lexical analysis;	4
Implementation of recursion in development of a number software;	6
Block diagrams	4
Finite automaton	4

Learning outcomes and assessment

Learning outcomes	Assessment methods
Students are able to independently study the grammatical qualities and develop their own grammar and determine which forms of analysis can be implemented in the software	A test in which the student develops grammar, for example, a grammar that treats arithmetical operations as a priority.
Students can use compilers specific algorithms for other tasks.	Homework in which the student performs, for example, the inclusion and retrieval of data in symbol tables.
Students can compare the performance of different types of analyzers.	Homework, in which the student evaluates the performance of analyzers in the analysis of the magnitude of the number of operations to be performed.

Study subject structure

Part	CP	Hours per Week			Tests		
		Lectures	Practical	Lab.	Test	Exam	Work
1.	2.0	1.0	0.0	1.0		*	