COURSE UNIT DESCRIPTION

Course unit title	Course unit code
Procedural programming	

Annotation
Course focuses on providing a strong foundation upon which further study of computer science can be built. It addresses various elements related (but not limited) to programming as well as general concepts related to information processing and computing. Alongside fundamental and technical knowledge, there is strong emphasis of developing general skills, including (and not limited to) communication and teamwork.

Lecturer(s)	Department where the course unit is delivered
Coordinator: Irmantas Radavičius **Other lecturers:**	Faculty of Mathematics and Informatics Vilnius University

Cycle	Type of the course unit
First	Compulsory

Mode of delivery	Semester or period when the course unit is delivered	Language of instruction
Mixed	1st semester	English, Lithuanian

Prerequisites
Prerequisites: none

Number of credits allocated	Workload of the student	Contact hours	Individual work
5	134	68	66

Purpose of the course unit: program competences to be developed

Purpose of the course unit – to provide students with knowledge and skills in procedural programming, using C programming language, and develop habits of disciplined development.

Generic competences:

- will be able to organise their own work independently (GK 1.3)
- will recognize of the need for, and engage in life-long learning (GK 2.1)
- will be able to independently acquire new knowledge, methods and tools and apply them in practice. (GK 2.3)
- will understand professional and ethical responsibility (GK 3.1)

Specific competences:

- will be able to apply mathematical foundations, knowledge of science and engineering, computer science theory, and algorithmic principles in software systems development (SK 4.2)
- will be able to reason at abstract level, to use formal notation, to prove the correctness, and to apply formalisation and specification for real-world problems (SK 4.3)
- will be able to use existing hardware, software and application systems, to identify, understand and apply the promising technologies (SK 6.3)
- will be aware of project management, quality assurance, and process improvement practices and develop abilities to apply them (SK 6.6.)

Learning outcomes of the course unit: students will be able to	Teaching and learning methods	Assessment methods
Demonstrate knowledge and skills in procedural programming	Lectures Assignments Individual work	Assignments Homework Tests Exam (written)
Efficiently and safely apply pointers to solve problems		
Create modules and apply principles of modular programming		

Use C-like syntax and learn new programming languages related to C more efficiently		
Understand the need for and use disciplined programming		

Course content: breakdown of the topics	**Contact hours**						**Individual work: time and assignments**	
	Lectures	Tutorials	Seminars	Practice	Laboratory work	**Contact hours**	**Individual work**	**Assignments**
Course overview. Introduction. Programming. Programming languages, their features. C language, history. Programming tools. First program	2				2	**4**	**4**	Individual reading Homework
Basic elements of programming languages. Data types. Values, variables. Operations, operators, operands, expressions. Input and output.	2				2	**4**	**4**	
Control structures. Decision making. Conditional and loop statements. Structured programming.	2				2	**4**	**4**	
One-dimensional and multi-dimensional arrays. Operations with arrays. Text strings.	2				2	**4**	**4**	
Streams. Operations with files. User interface. Data validation. Formatted input and output.	2				2	**4**	**4**	
Addresses and pointers. Pointer arithmetic. Static and dynamic arrays. Dynamic memory allocation.	2				2	**4**	**4**	
Functions, declarations, and definitions. Function calls, recursion. Local and global variables. Parameter passing. Functional decomposition, procedural programming.	2				2	**4**	**4**	
Primitive and derived data types. Type conversion, operator precedence, evaluation order. Structured data types. Data structures.	2				2	**4**	**4**	
Testing. Errors, their types, debugging and prevention. Security loopholes, safe coding practices.	2				2	**4**	**4**	
C preprocessor. Phases of translation. Preprocessor directives. Macroses. Conditional compilation.	2				2	**4**	**4**	
C system. Compiler options. Make tool. Header files. Multi-file programs. Modular programming.	2				2	**4**	**4**	
Libraries. C standard library.	2				2	**4**	**4**	
Storage classes. Variable lifecycle, scope. Constants.	2				2	**4**	**4**	
Algorithms, pseudocodes. Creating, evaluating, and optimizing algorithms. Searching and sorting problems.	2				2	**4**	**4**	
Positional numbering systems. Bit management.	2				2	**4**	**4**	
Grammars. Definitions of programming languages. Coding standards.	2				2	**4**	**4**	
Feedback						**2**	**0**	

Preparation for the exam							**2**	
Exam						**2**		
Total	**32**				**32**	**68**	**66**	

Assessment strategy	**Weight, perc.**	**Deadline**	**Assessment criteria**
Assignments	10	During the semester	During the semester, students complete various assignments and receive up to 1 (one) point for doing this. Based on the requirements and evaluation criteria for each specific assignment, students can get a grade from 0 to 10, which gets multiplied by a specific coefficient for each task and gets added to the total sum accumulated.
Homework	40	During the semester	During the semester students get 4 (four) homework assignments. They get evaluated based on the correctness of the solution, delivery within the deadlines, quality, as well as knowledge and skills demonstrated. First homework assignment gives up to 0.5 points; second and third up to 1.0 points; fourth – up to 1.5 points. Students are allowed to take the exam only after completing 3 assignments and getting a total of at least 2 points (out of 4).
Tests	0	During the semester	During the semester students get up to 8 tests, to check the minimal knowledge requirements for this course. Students are allowed to take the exam only after passing every such test. Tests do not give points, all questions have equal value, and to pass, a student is required to correctly answer more than half of the questions. The lecturer can allow everyone to re-take the chosen test, based on the results.
Bonuses	10	During the semester	During the semester, lecturers can give up to 1 (one) bonus point, based on the effort and results demonstrated by the student.
Exam (written)	50	January	Students are allowed to take the exam after passing all tests, and completing at least 3 homework assignments, as well as getting at least 2 points (out of 4). Exam consists of various questions to check the knowledge of the student, their ability to read and write code, and other problems. Students can get up to 5 points. To pass, a student is required to get at least 2 points (out of 5), otherwise they get a grade not bigger than 4. Students who get more than 10 points is given 10.
Extern		The student can repeat the course externally, if before they have participated fully and they accept the previously collected number of points. In this case, the points get accounted for and the student only repeats the exam. The student who is taking the course unit externally must inform the lecturer in the beginning of the semester and get the written consent with the above-mentioned number of points confirmed. If the student has not collected the minimal number of points required to pass, or the number of points collected does not suit the student, the subject cannot be repeated externally.	

Author	**Publishing year**	**Title**	**Number or volume**	**Publisher or URL**
Required reading				
Brian W. Kernighan, Dennis M. Ritchie	1988	The C Programming Language.	2nd ed.	Prentice Hall
Saulius Ragaišis	2007	Personal software development process (in Lithuanian)		https://klevas.mif.vu.lt/~ragaisis/PSP2007/Asmeninis.programu.kurimo.procesas.pdf
Recommended reading				
K. N. King.	2008	C Programming: A Modern Approach.	2nd ed.	W. W. Norton & Company
Paul J. Deitel, Harvey M. Deitel	2013	C – How to Program.	7th ed.	Prentice Hall
Stephen G. Kochan	2004	Programing in C.	3rd ed.	Sams Publishing
Robert C. Martin	2009	Clean Code		Pearson Education, Inc
Andy Hunt,	1999	The Pragmatic Programmer		Addison Wesley

Dave Thomas				
W.S. Humphrey	1997	Introduction to the Personal Software Process		Addison-Wesley
Nick Parlante	2003	Essential C		http://cslibrary.stanford.edu/101/
Anand Mehta	1995	A Crash Course in C		http://www.mattababy.org/~belmonte/Teaching/CCC/handouts.pdf