COURSE UNIT (MODULE) DESCRIPTION

Course unit (module) title	Code
PROGRAMMING LANGUAGES AND OBJECT-ORIENTED PROGRAMMING	

Academic staff	Core academic unit(s)
Coordinating: assoc. prof. Vytautas Rudžionis **Other:**	Kaunas Faculty Institute of Social Sciences and Applied Informatics Muitinės g. 8, LT–44280 Kaunas

Study cycle	Type of the course unit
First	Mandatory

Mode of delivery	Semester or period when it is delivered	Language of instruction
In class	2 semester	English

Requisites	
Prerequisites: Students must have mastered the basics of algorithm theory and data structures, computer architecture, and programming.	**Co-requisites (if relevant):**

Number of ECTS credits allocated	Student's workload (total)	Contact hours	Individual work
5	133	48	85

Purpose of the course unit		
Develop the ability to program in high-level programming languages, develop the ability to understand and analyze their organizational principles and programming paradigms (procedural and object-oriented programming); develop the ability to understand, analyze, and apply the principles of object-oriented programming and the principles of graphical interface design; develop the ability to create simple software applications and generate program code.		
Learning outcomes of the course unit	**Teaching and learning methods**	**Assessment methods**
Will be able to transform an algorithm presented in program code, be able to select the most effective software tools for a given algorithm.	Lectures, tutorials, laboratory work, problem solving, active learning methods (algorithm analysis, program prototype writing, system prototype design)	Laboratory work, laboratory work defenses, independent systems analysis, problem solving, test work, exam
Will be able to apply object-oriented programming principles to the given algorithm, be able to select more effective program development methods.	Lectures, tutorials, laboratory work, problem solving, active learning methods (algorithm analysis, program prototype writing, system prototype design)	Laboratory work, laboratory work defenses, independent systems analysis, problem solving, test work, exam
Will be able to write medium-complexity programs, evaluate program compliance with specifications, and know some of the best programming practices.	Lectures, tutorials, laboratory work, problem solving, active learning methods (algorithm analysis, program prototype writing, system prototype design)	Laboratory work, laboratory work defenses, independent systems analysis, problem solving, test work, exam

Content	Contact hours							Individual work: time and assignments	
	Lectures	Tutorials	Seminars	Workshops	Laboratory work	Internship	**Contact hours, total**	**Individual work**	**Tasks for individual work**
1. Programming languages. Classification of programming languages. Low-level and high-level programming languages. Basic components of programming languages. GUI.	2			4			6	10	Program writting
2. The most popular programming languages: C/C++, Java, C#. Specific features of these programming languages.	2			2			**4**	10	Program writting
3. Fundamentals of object-oriented programming: the concept of an object; the concept of a class; object-oriented programming; encapsulation; inheritance; advantages of object-oriented programming.	4			8			12	20	Program writting
4 Fundamentals of object-oriented programming (2): polymorphism; templates; virtual functions; object instances.	4			8			12	18	Program writting and code analysis
5. Principles of creating a graphical user interface: principles of graphical interfaces; window; components; events and their processing; parallel processing of multiple tasks.	2			4			6	10	Program code analysis, principles of software system development, preparation for an exam.
6. Principles of automatic code generation: advantages of code reuse; automatic code generation; limitations of automatic code generation; automatic code generation tools.	2			4			6	10	Program code analysis, principles of software system development, preparation for an exam
Exam				2			**2**	**5**	Preparation for an exam
Total	**16**			**32**			**48**	**85**	

Assessment strategy	Weight %	Deadline	Assessment criteria
Control work (K1)	15	On a predefined date	The student is given a task to write a program within 1 hour. It is graded on a 10-point scale according to the following criteria: implementation of functional requirements; accuracy of algorithm implementation; accuracy of program code.
Control work (K2)	15	On a predefined date	The student is given a task to write a program within 1 hour. It is graded on a 10-point scale according to the following criteria:

			implementation of functional requirements; accuracy of algorithm implementation; accuracy of program code.
Individual program preparation (S)	20	On a predefined date	Students are given an assignment to create a decision algorithm, define functional requirements, and write a program prototype in the programming language of their choice. They are graded on a 10-point scale according to the following criteria: accuracy of algorithm implementation; accuracy of program code; efficiency of program code; implementation of functional requirements; ability to modify the code; program reliability; program functionality The use of code generation tools is prohibited.
Exam	50	On a predefined date	The test consists of 10 closed-ended questions (of varying difficulty, ranging from understanding algorithms to knowledge of theoretical foundations), each worth one point. The scoring is as follows: each question is worth one point. Exam scores are weighted with a coefficient of 0.5 in the final grade.
Final mark: 0.15*K1+0.15*K2+0.20*S+0.50*E Extern exam assessment strategy: Not applicable Using of AI tools not permited if not stated otherwise by lecturer			

Author (-s)	Publishing year	Title	Issue of a periodical or volume of a publication	Publishing house or web link
Required reading				
Mak R.	2024	Object-Oriented Software Design in C++		New York: Manning Publications
Harwani B.M.	2015	Learning object-oriented programming in C# 5.0		Boston, MA : Cengage Learning PTR
McLaughlin B., Police G.	2008	Head-First Object Oriented Analysis and Design		O'Reilly
Recommended reading				
Wagner B.	2010	Effective C#		New York: Addison-Wesley. (https://livres.ycharbi.fr/Livres/ebook%20informatique/Informatique/Langages/C%23%2C%20.Net%20Framework/Effective%20C%23%2C%2050%20Specific%20Ways%20to%20Improve%20your%20C%23%20%282nd%20ed%2C%202010%29%20-%20%5BAddison-Wesley%5D%20-%20Bill%20Wagner.pdf)
Gamma E., Helm R., Johnsin R., Vlissides J.	2012	Design Patterns: Elements of Reusable Object-Oriented Software		New York: Wiley. (http://www.uml.org.cn/c%2B%2B/pdf/DesignPatterns.pdf)

Horstmann C.	2021	Big C++. Late Objects		New York: Wiley. (https://horstmann.com/bigcpp/bigcpp1.html)
Liberty J., McDonald B.	2008	Learning C#		Boston: O'Rilley.
Lafore R.	2002	Object-Oriented Programming in C++		https://docs.google.com/file/d/0B21HoBq6u9TsUHhqS3JIUmFuamc/view?resourcekey=0-MYlet9RIjEukd6CvLEHUbw

NOTE: *Including Open Educational Resources in the reading list is recommended*