Subject code	ECTS credits
INF1007	6

Course title in Lithuanian

OBJEKTINIS PROGRAMAVIMAS

Course title in English

OBJECT ORIENTED PROGRAMMING

Short course annotation in Lithuanian (up to 500 characters)

Dalykas skirtas C++ objektinio programavimo priemonių parengimo ir naudojimo principų studijoms. Studentai supažindinami su GITHUB repozitorija, rekursija, rodykle, simbolių eilučių, abstrakčių tipų sąvokomis ir jų realizavimu bei panaudojimu. Suteikiami objektinio programavimo pagrindai, mokinama formuoti objektinius programų modelius. Analizuojamos klasių aprašymo priemonės, klasių vidinės struktūros paslėpimas, išorinės sąsajos aprašymo priemonės, savybių paveldėjimas klasių šeimose ir jų polimorfiškumas, klasių kompozicijos, kritinių situacijų kontrolės bei šabloninio programavimo priemonės, susietų sąrašų tvarkymo klasės.

Short course annotation in English (up to 500 characters)

Subject is suited to learn C++ programming language as object-oriented programming tools, and to get use it in simple examples. Understands GitHub repositories. Students are introduced to recursion, pointers, string data type, abstract data type, and concepts of their realization. Basics of object-oriented programming are provided, developing of the object models and applications are introduced. Overview of classes, objects, methods, descriptions for hiding the internal structure and external links are given. Characteristics of inheritance, polymorphism, composition classes, critical situations control, planning of graphical user interface, programming of it are explained.

Prerequisites for entering the course

Programming Fundamentals

Course aim

Understand of designing of advanced algorithms, get introduced to object oriented programming.

Links between course outcomes, criteria of learning achievement evaluation, study methods and methods of learning achievement assessment

No	Course outcomes	Criteria of learning achievement evaluation	Study methods	Methods of learning achievement assessment
1.	Choose and apply software to solve practical problems.	Ability to distinguish the programming languages and compound technologies. Reasonably understands significance of exceptions and influence for software stability.	Practical works; Giving inter- pretations and illustra- tions through visual material; Reviewing material;	Observations of students presen- tations, individual practical activities. Evaluation of semester work, written reports, classroom tests, written mid-term and final examinations.
2.	Understand GitHub repositories	Understanding the GitHub repositories. Student demonstrates skills in software developing flow, systems and applying the tools.	Giving inter- pretations and illustra- tions through visual material. Practical works.	Evaluation and analysis of the practical works.
3.	Understanding Practical benefits of Recursion.	Knows development and work-flow of recursion algorithms.	Building a problem set. Practical works;	Observations of students works, presentations, individual practical activities.

		Student presents the practical works to lecture and their colleagues.		Evaluation of semester work, written reports, classroom tests, written mid-term examination.
4.	Applying the Pointers.	Understand the meaning of reference to the object, can distinguish the storing of variable and storing memory address. Student presents the practical works to lecture and their colleagues.	Building a problem set. Practical works;	Observations of students works, presentations, individual practical activities. Evaluation of semester work, written reports, classroom tests, written mid-term examination.
5.	Understand object-oriented concepts.	Knows the keywords: class, object, method, inheritance, hierarchy. Student presents the practical works to lecture and their colleagues.	Building a problem set. Practical works;	Observations of students works, presentations, individual practical activities. Evaluation of semester work, written reports, classroom tests, written final examination.
6.	Interchange of objects with static and dynamic data fields.	Understanding and distinguishing the methods of storing data in computers' memory. Student presents the practical works to lecture and their colleagues.	Building a problem set. Practical works;	Observations of students works, presentations, individual practical activities. Evaluation of semester work, written reports, classroom tests, written final examination.
7.	Provide knowledge on building of an algorithm, developing a program, providing an analysis of working, or non-working program.	Reasonably understands the significance of exceptions and influence for software stability. Student demonstrates skills in developing systems and applying the tools. Student presents the practical works to lecture and their colleagues.	Building a problem set. Giving inter-pretations and illustrations through visual material; Practical works; Reviewing material;	Evaluation of oral presentation and analysis of the practical works. Observations of students works, presentations, individual practical activities. Evaluation of semester work, written reports, classroom tests, written mid-term and final examinations.
8.	Design and developing of Graphical user interface	Understanding the fundamentals of building and GUI. Student presents the practical works to lecture and their colleagues.	Building a problem set. Practical works;	Observations of students works, presentations, individual practical activities. Evaluation of semester work, written reports, classroom tests,

9.	Choose and apply suitable tools, interpret the results.	The ability to use received knowledge in other university courses. Student demonstrates skills in developing systems and applying the tools.	Giving inter- pretations and illustra- tions through visual material; Reviewing material;	Observations of students presentations, individual practical activities. Evaluation of oral presentation and analysis of the practical works, written reports, classroom tests, written mid-term and final examinations.

Links between study programme outcomes and course outcomes

Study programme outcomes	Running number of course outcome								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Know and comprehend the needs and importance of information technologies in study process, also be able to apply programming knowledge and skills, data structures and modelling	+	+	+	+	+	+	+		+
Identify the problem, collect and analyze real/theoretical data using various mathematical methods, tools and IT technologies	+	+			+	+	+	+	

Content

No	Content (topics)
1.	Differences of structured/functional and object oriented programming
2.	Structure data type. Recursion. Pointers.
3.	Object-oriented concept: Object-oriented programming: an Object, a Class, EncapsulationPolymorphism, Inheritance, Multiinheritance. Construction. Destruction. Namespaces. Virtual methods. Templates.
4.	Abstract data type. Designing a General Class Structure.
5.	Classes with the Dynamical data fields
6.	Exception handling
7.	GitHub
8.	User interface, graphical user interface modelling

Distribution of workload for students (contact and independent work hours)

Practicum	**75 hours**
Individual students work	**85 hours**
Total:	**160 hours**

Structure of cumulative score and value of its constituent parts

Final written exam (50%), mid-term written exam (17%), and assessments of laboratory (practical) work (33%).

Recommended reference materials

No.	**Publication year**	**Authors of publication and title**	**Publishing house**	**Number of copies in**		
				University library	*Self-study rooms*	*Other libraries*
Basic materials						
1.	2016	V.Barzdaitis „Objektinio progra- mavimo pagrindai“ -				Electronic papers, in distance learnig system: http://moodle.vdu.lt

		distance learning course				
2.	2013	C++ Programming Language OOP				https://www3.ntu.edu.sg/home/ehchua/ programming/cpp/cp3_OOP.html
3.	2008	A.Vidžiūnas „C++ ir objektinis programavimas“	10	5		
Supplementary materials						
1	2016	Visual Studio Quick Reference Guidance	SlideShare		Free resources on SlideShare: *https://vsarquickguide.codeplex.com*	
2	2016	Visual C++ Developer Center			Free resources on Internet: https://msdn.microsoft.com/en-us/vstudio/aa718325.aspx	
3	2016	CPP programming tutorials, best practice examples, working examples, debuging instructions			http://www.bogotobogo.com/ cplusplus/cpptut.php http://www.cplusplus.com http://www.learncpp.com/	
4		Free forums resources: best news, issues solving solutions.			http://stackoverflow.com/questions/ 388242/ the-definitive-c-book-guide-and-list https://www.quora.com/What-are-the-best-C++-books	

Course programme designed by

Lect. Vytautas Barzdaitis