Syllabus

1. Information regarding the programme

1.1 Higher education	Babeş-Bolyai University
institution	
1.2 Faculty	Faculty of Mathematics and Computer Science
1.3 Department	Department of Computer Science
1.4 Field of study	Mathematics
1.5 Study cycle	Bachelor
1.6 Study programme /	Mathematics and Computer Science
Qualification	

2. Information regarding the discipline

2.1 Name of the di	scipli	ne (en)	Object Oriented Programming				
(ro)			Pro	ogramare orientată o	biect	t	
2.2 Course coordinator		Lect. PhD Diana Laura Borza					
2.3 Seminar coordinator			Le	Lect. PhD Diana Laura Borza			
2.4. Year of study	1	2.5	2	2.6. Type of	E	2.7 Type of	Compulsory
Semester			evaluation		discipline		
2.8 Code of the MLE5006							
discipline							

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	5	Of which: 3.2 course	2	3.3	1 sem
				seminar/laboratory	2 lab
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6	42
				seminar/laboratory	
Time allotment:					
Learning using manual, course support, bibliography, course notes					
Additional documentation (in libraries, on electronic platforms, field documentation)					
Preparation for seminars/labs, homework, papers, portfolios and essays					
Tutorship					
Evaluations					
Other activities:					
1					

3.7 Total individual study hours	80
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	Fundamentals of programming
4.2. competencies	Average programming skills in a high-level programming
	language

5. Conditions (if necessary)

5.1. for the course	· Class room with projector
5.2. for the seminar /lab	· Laboratory with computers, having a C++ compiler, a C++ IDE
activities	(preferably Visual Studio) and Qt library installed

6. Specific competencies acquired

or specific cor	inferencies acquired
Professio nal compete ncies	 C1.1 Description of programming paradigms and of language specific mechanisms, as well as identification of syntactic and semantic differences. C1.2 Explanation of existing software applications, on different levels of abstraction (architecture, classes, methods) using adequate basic knowledge. C1.3 Elaboration of adequate source codes and testing of components in a given programming language, based on some given specifications. C1.4 Testing applications based on testing plans. C1.5 Developing units of programs and corresponding documentations.
Transver sal compete ncies	 CT1 Application of efficient and rigorous working rules, manifest responsible attitudes towards the scientific and didactic fields, respecting the professional and ethical principles. CT2 Use of efficient methods and techniques for learning, information, research and development of abilities for knowledge exploitation, for adapting to the needs of a dynamic society and for communication in Romanian as well as in a widely used foreign language.

7. Objectives of the discipline (outcome of the acquired competencies)

7. Objectives of the discipline	7. Objectives of the discipline (outcome of the acquired competencies)			
7.1 General objective of the	· To understand the concepts of the object-oriented programming			
discipline	paradigm and to design object-oriented solutions of small/medium			
	scale problems, using C++ and Qt.			
7.2 Specific objective of the	· To demonstrate the differences between traditional imperative design			
discipline	and object-oriented design.			
	· To explain class structures as fundamental, modular building blocks.			
	To understand the role of inheritance, polymorphism, dynamic binding			
	and generic structures in building reusable code.			
	To explain and to use defensive programming strategies, employing			
	formal assertions and exception handling.			
	· To design user- interfaces and write small/medium scale C++			
	programs using Qt.			
	· To use classes written by other programmers and third-party libraries			
	when constructing their systems.			

8. Content

8.1 Course	Teaching methods	Remarks
1. C/C++ introduction (basic elements of	· Interactive exposure	
C/C++ programming language, data types,	· Explanation	
constant variables, scope and lifetime of the	· Conversation	
variables, statements, functions: declaration	· Examples	
and definition, overloading functions).	· Didactical	
	demonstration	
2. Modular programming in C/C++	· Interactive exposure	
(functions, formal and actual parameters,	· Explanation	
pointers and memory management, the stack	· Conversation	
	· Examples	

and the help, pointers to functions, header files, modular programming, libraries). 3. Object oriented programming in C++ (introduction to object oriented programming, object oriented programming factors, abstraction, encapsulation, classes and objects, access modifers, object creation and destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL array, vector, list, stack, heap, map, sed), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std::unique ptr, std::shared ptr, std::weak ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GTU components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming I (Model View events, signals and slots in Qt), using			
3. Object oriented programming in C++ (introduction to object oriented programming, object oriented programming features, abstraction, encapsulation, classes and objects, access modifiers, object creation and destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UMI. class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iteratory, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Intitalization (RAII), smart pointers, move semantics, smart pointers in STL: staticulation (Date to a demonstra			
(introduction to object oriented programming, object oriented programming features, abstraction, encapsulation, classes and objects, access modifiers, object creation and destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++, The C++ Standard Template Library (function templates, class templates, containers in STL array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception and extraction operators, move senantics, smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::wak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming I (Model View Interactive exposure			
object oriented programming features, abstraction, encapsulation, classes and objects, access modifiers, object creation and destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++, The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), licrators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::wak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming I (model View interestion and territorie exposure interestion examples bidactical demonstration 12. Event driven programming II (model View interestion extension) 13. Event driven programming II (model View interestion interestive exposure	3.	• • • • • •	±
abstraction, encapsulation, classes and objects, access modifiers, object creation and destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, container in STL: stary, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition 1s Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::wak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure			
destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exceptions after odde, bandling, exceptions after odde, bandling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std::mique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure			
destruction, operator overloading, static and friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downeasting, abstract classes, UMI, class diagrams and relations). 6. Templates in C++, The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Interactive exposure Explanation Conversation Interactive exposure Explanation Interactive exposure Explanation Conversation Interactive exposure Explanation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Explanation Conversation Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Explanation Conversation Interactive exposure Explanation Conversation Interactive exposure Explanation Interactive exposure Explanation Conversation Interactive exposure Interactive exposure Interactive exposure Intera			=
friend elements). 4. Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std: unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure interactive exposure interactive exposure interaction exposure interactive exposure interaction expanding interactive exposure inte			
derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++, The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std: unique_ptr, std::shared_ptr, of Ultimized and instruments, Qt GU components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, event) events driven and struction demonstration 11. Event driven programming II (Model View Interactive exposure Interactive exposure Explanation Conversation Explanation Conversation Explanation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Interactive exposure Explanation Conversation Explanation Conversation Explanation Conversation Explanation Conversation Explanation Conversation Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Explanation Conversation Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Explanation Conversation Explanation Interactive exposure Expla			demonstration
derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, continued by the components, Layout management, design interfaces using Qt Designer). 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Explanation 12. Event driven programming II (Model View Interactive exposure Explanation 13. Event driven programming II (Model View Interactive exposure		,	
method overriding, inheritance and polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAHI (Resource Acquisition Is Initialization (RAHI), smart pointers, move semanties, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Conversation Interactive exposure Explanation	4.		· Interactive exposure
polymorphism). 5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction poperators, overloading insertion and extraction is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GIU components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Interactive exposure Explanation Conversation 12. Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration		derived classes, Liskov substitution principle,	Explanation
Didactical demonstration		method overriding, inheritance and	· Conversation
5. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction properators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition is Initialization (RAH), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Explanation Conversation Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Interactive exposure Explanation Interactive exposure Interactive exposu		polymorphism).	· Examples
S. Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). UML class diagrams and relations).			· Didactical
virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAH (Resource Acquisition Is Initialization (RAH), smart pointers, move semantics, smart pointers, move semantics, smart pointers, move semantics, smart pointers, tiles distingue ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure) 12. Examples 13. Explanation 14. Conversation 15. Examples 16. Conversation 16. Examples 17. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction 12. Examples 13. Event driven programming II (Model View Interactive exposure)			demonstration
upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View	5.	Polymorphism (static and dynamic binding,	· Interactive exposure
upcasting and downcasting, abstract classes, UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View			- I
UML class diagrams and relations). 6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure			±
6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View 11. Event driven programming II (Model View 12. Interactive exposure Explanation Conversation Examples Didactical demonstration 12. Interactive exposure Explanation Conversation Examples Didactical demonstration 13. Interactive exposure Explanation Conversation Examples Didactical demonstration 14. Event driven programming II (Model View 15. Interactive exposure Explanation Conversation Examples Didactical demonstration 16. Interactive exposure Explanation Conversation Examples Didactical demonstration 16. Event driven programming II (Model View 16. Interactive exposure Explanation Conversation Examples Didactical demonstration 16. Interactive exposure Explanation Conversation Examples Didactical demonstration			· Examples
6. Templates in C++. The C++ Standard Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View) 11. Event driven programming II (Model View) Interactive exposure Explanation Conversation Explanation Conversati		,	=
Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View) Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration			demonstration
Template Library (function templates, class templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View) Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration	6.	Templates in C++. The C++ Standard	
templates, containers in STL: array, vector, list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAHI (Resource Acquisition Is Initialization (RAHI), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure		•	±
list, stack, heap, map, set), iterators, STL algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration		• • • • • • • • • • • • • • • • • • • •	
algorithms, lambda functions. 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure			
demonstration 7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure			<u>-</u>
7. Streams and exception handling (input output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View) Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Interactive exposure Interactive exposure Explanation Interactive exposure Interactive exposure Interactive exposure		argoriumis, iamoda ranovions.	
output streams, insertion and extraction operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration	7	Streams and exception handling (input	
operators, overloading insertion and extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAHI (Resource Acquisition Is Initialization (RAHI), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Conversation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure	, ,	• • • • • • • • • • • • • • • • • • • •	• I
extraction operators, formatting, manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure			
manipulators, flags, text files, exception handling, exception safe code). 8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Didactical demonstration Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure			
handling, exception safe code). 8. Resource management and RAHI (Resource Acquisition Is Initialization (RAHI), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Examples Didactical demonstration 12. Event driven programming II (Model View Interactive exposure		· · · · · · · · · · · · · · · · · · ·	*
8. Resource management and RAII (Resource Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Interactive exposure Examples Didactical demonstration Interactive exposure Examples Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Explanation Interactive exposure			
Acquisition Is Initialization (RAII), smart pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Interactive exposure Explanation Interactive exposure Interactive exposure Examples Interactive exposure	Q		
pointers, move semantics, smart pointers in STL: std::unique_ptr, std::shared_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Interactive exposure Explanation Interactive exposure	0.	· · · · · · · · · · · · · · · · · · ·	<u> </u>
STL: std::unique_ptr, std::shared_ptr, std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View SExamples Conversation Examples Didactical demonstration Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Examples Interactive exposure Interactive exposure Explanation Interactive exposure Interactive exposure		* * * * * * * * * * * * * * * * * * * *	±
std::weak_ptr) 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming I (Model View Didactical demonstration Examples Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure		1 / 1	
demonstration 9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View demonstration Interactive exposure Examples Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Explanation Interactive exposure Explanation Interactive exposure			=
9. Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View 12. Interactive exposure Explanation 13. Interactive exposure Explanation 14. Event driven programming II (Model View 15. Interactive exposure Explanation 16. Examples Exampl		stuweak_pu)	
installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Explanation Conversation Interactive exposure Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Interactive exposure Examples Didactical demonstration Interactive exposure		Cuanhical Heav Interfered (Ot Teelleit)	
GUI components, Layout management, design interfaces using Qt Designer). 10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View Conversation Examples Explanation Conversation Examples Didactical demonstration Interactive exposure Explanation Examples Didactical demonstration Interactive exposure	9.	•	±
design interfaces using Qt Designer). Didactical demonstration 10. Event driven programming I (callbacks, events, signals and slots in Qt). Examples Examples Interactive exposure Explanation Conversation Examples Didactical demonstration 11. Event driven programming II (Model View Interactive exposure			
- Didactical demonstration 10. Event driven programming I (callbacks, events, signals and slots in Qt). - Explanation - Conversation - Examples - Didactical demonstration 11. Event driven programming II (Model View - Interactive exposure			
demonstration 10. Event driven programming I (callbacks, events, signals and slots in Qt). Explanation Conversation Examples Didactical demonstration 11. Event driven programming II (Model View demonstration Interactive exposure		design interfaces using Qt Designer).	=
10. Event driven programming I (callbacks, events, signals and slots in Qt). 11. Event driven programming II (Model View 12. Interactive exposure 13. Event driven programming II (Model View 14. Interactive exposure 15. Event driven programming II (Model View 16. Interactive exposure 17. Interactive exposure 18. Interactive exposure 18. Interactive exposure 19. Explanation 10. Explanation 10. Event driven programming II (Model View 10. Interactive exposure 11. Interactive exposure			
events, signals and slots in Qt). Explanation Conversation Examples Didactical demonstration 11. Event driven programming II (Model View Interactive exposure		B (1)	
Conversation Examples Didactical demonstration 11. Event driven programming II (Model View Interactive exposure	10.		±
Examples Didactical demonstration 11. Event driven programming II (Model View Interactive exposure		events, signals and slots in Qt).	
Didactical demonstration 11. Event driven programming II (Model View Interactive exposure			
demonstration 11. Event driven programming II (Model View Interactive exposure			
11. Event driven programming II (Model View Interactive exposure			
Controller, Models and Views in Qt, using Explanation	11.	• • • • • • • • • • • • • • • • • • •	±
		Controller, Models and Views in Qt, using	· Explanation

predefined models, implementing custom models).	ConversationExamplesDidactical demonstration
12. Design patterns I (creational, structural, behavioral patterns, examples, singleton, factory method, adapter pattern).	 Interactive exposure Explanation Conversation Examples Didactical demonstration
13. Design patterns II (façade pattern, observer pattern, strategy pattern, case study application and examples).	 Interactive exposure Explanation Conversation Examples Didactical demonstration
14. Revision (revision of the most important topics covered by the course, examination guide).	 Interactive exposure Explanation Conversation Examples Didactical demonstration

Bibliography

- 1. B. Stroustrup. *The C++ Programming Language*, Addison Wesley, 1998.
- 2. Bruce Eckel. *Thinking in C++*, Prentice Hall, 1995.
- 3. A. Alexandrescu. *Programarea moderna in C++: Programare generica si modele de proiectare aplicate*, Editura Teora, 2002.
- 4. S. Meyers. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)*, Addison-Wesley, 2005.
- 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
- 6. B. Stroustrup. A Tour of C++, Addison-Wesley, 2013.
- 7. C++ reference (http://en.cppreference.com/w/).
- 8. Qt Documentation (http://doc.qt.io/qt-5/).
- 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Longman Publishing, 1995.

8.2 Seminar	Teaching methods	Remarks
Simple problems in C. Functions. Structures, enums and arrays.	Interactive exposureExplanationConversation	The seminar is structured as a 2 hour class, every 2 weeks.
2. Modular programming.	Interactive exposureExplanationConversation	
3. Classes. Operator overloading. User-defined objects as class data members.	Interactive exposureExplanationConversation	
4. Inheritance. Polymorphism. Templates.	Interactive exposureExplanationConversation	
5. Files, exceptions. STL containers, iterators, algorithms.	Interactive exposureExplanationConversation	

6. Graphical User Interfaces.	· Interactive exposure
	· Explanation
	· Conversation
7. Implementation based on UML diagrams.	· Interactive exposure
Design patterns.	Explanation
	· Conversation

Bibliography

- 1. B. Stroustrup. *The C++ Programming Language*, Addison Wesley, 1998.
- 2. Bruce Eckel. *Thinking in C++*, Prentice Hall, 1995.
- 3. A. Alexandrescu. *Programarea moderna in C++: Programare generica si modele de proiectare aplicate*, Editura Teora, 2002.
- 4. S. Meyers. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)*, Addison-Wesley, 2005.
- 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
- 6. B. Stroustrup. A Tour of C++, Addison-Wesley, 2013.
- 7. C++ reference (http://en.cppreference.com/w/).
- 8. Qt Documentation (http://doc.qt.io/qt-5/).
- 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Longman Publishing, 1995.

8.3 Laboratory	Teaching methods	Remarks
1. Environment setup (installing a C++ compiler and an IDE). C/C++ basics.	ExplanationConversation	The laboratory is structured as weekly 2 hour classes.
2. Introductory problems (in C).	ExplanationConversation	
3. Feature-driven software development process. Layered architecture. Test driven development. Modular programming	ExplanationConversation	
4. Classes and objects in C++. Copy constructors, assignment operators, destructors.	ExplanationConversation	
5. Inheritance. Method overriding.	ExplanationConversation	
6. Inheritance and polymorphism. Virtual methods.	ExplanationConversation	
7. Laboratory test.	Practical test	
8. STL containers, iterators and algorithms.	ExplanationConversation	
9. Streams, overloading the insertion and extraction operators, persistence.	ExplanationConversation	
10. Exception handling. Testing.	ExplanationConversation	
11. Qt Graphical User Interfaces I.	ExplanationConversation	
12. Qt Graphical User Interfaces II. Signals and slots in Qt.	ExplanationConversation	
13. Design patterns.	ExplanationConversation	
14. Laboratory test.	Practical test	

Bibliography

- 1. B. Stroustrup. *The C++ Programming Language*, Addison Wesley, 1998.
- 2. R. Gilberg. C++ Programming: An Object-Oriented Approach, McGraw-Hill Education, 2019
- 3. A. Alexandrescu. *Programarea moderna in C++: Programare generica si modele de proiectare aplicate*, Editura Teora, 2002.
- 4. S. Meyers. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)*, Addison-Wesley, 2005.
- 6. B. Stroustrup. A Tour of C++, Addison-Wesley, 2013.
- 7. C++ reference (http://en.cppreference.com/w/).
- 8. Qt Documentation (http://doc.qt.io/qt-5/).
- 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Longman Publishing, 1995.
- 10. Bruce Eckel. *Thinking in C++*, Prentice Hall, 1995.

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- · The course respects the ACM Curricula Recommendations for Computer Science studies.
- The course exists in the studying program of all major universities in Romania and abroad.
- The content of the course is considered by the software companies as important for average object-oriented programming skills.

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct C++ programs.	Written examination (regular session).	60%
10.5 Seminar/lab activities	Ability to design, implement, test and debug a C++ program with a graphical user interface.	Practical evaluation. Two tests during the semester.	20%
10.6 Minimum performance	Project.	Design, implementation and testing of a small-medium application that uses a 3-tier architecture. Documentation	20%

10.6 Minimum performance standards

- □ Students must prove that they acquired an acceptable level of knowledge and understanding of the core concepts taught in the class, that they are capable of using this knowledge in a coherent form, that they have the ability to establish certain connections and to use the knowledge in solving small/medium scale problems using object-oriented programming in C++.
- □ Successfully passing the examination is conditioned by a minimum grade of 5 (no rounding) for the laboratory practical test, the laboratory assignment and written examination.
- ☐ Attendance is mandatory for 5 seminar sessions and 12 laboratory sessions.

Date	Signature of course coordinator	Signature of seminar coordinator
02.10.2024	Lect. PhD. Diana Laura Borza	Lect. PhD. Diana Laura Borza
Date of approval	Signature of the head of department	