COURSE DESCRIPTION

Object oriented programming basics

Academic year 2026-2027

1. Programme-related data

1.1. Higher Education Institution	Babeș-Bolyai University
1.2. Faculty	Faculty of Mathematics and Computer Science
1.3. Department	Department of Computer Science
1.4. Field	Mathematics
1.5. Level of study	Bachelor
1.6. Degree programme / Qualification	Mathematics and Computer Science (double specialization)
1.7. Form of education	Full time

2. Course-related data

2.1. Course title	**Object Oriented Programming**			Course code	MLE5234
2.2. Course coordinator		Lect. PhD. Diana-Laura Borza			
2.3. Seminar coordinator		Lect. PhD. Diana-Laura Borza			
2.4. Year of study	1	2.5. Semester	2	2.6. Type of assessment	Exam
2.7. Course status		Compulsory		2.8. Course type	Core subject

3. Total estimated time (hours per semester of teaching activities)

3.1. Number of hours per week	**5**	of which: 3.2. course	**2**	3.3. seminar/ laboratory/ project	**3**
3.4. Total of hours in the curriculum	70	of which: 3.5. course	28	3.6. seminar/ laboratory	**42**
Time allocation for individual study (IS) and self-taught activities (ST)					**hours**
Learning from textbooks, course materials, bibliography, and notes (IS)					20
Additional research in the library, on subject-specific electronic platforms, and on-site					25
Preparing seminars/ laboratories/ projects, assignments, reports, portfolios, and essays					25
Tutoring (professional guidance)					5
Examinations					5
Other activities					
3.7. Total hours of individual study (IS) and self-taught activities (ST)				**80**	
3.8. Total hours per semester				**150**	
3.9. Number of credits				**6**	

4. Prerequisites (where applicable)

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

5. Specific conditions (where applicable)

5.1. course-related	Class room with projector
5.2. seminar/laboratory-related	Laboratory with computers, having a C++ compiler, a C++ IDE (preferably Visual Studio) and Qt library installed

6.1. Competencies resulting from the completion of the degree programme *(as referred to in the curriculum)*[1]

[1] The professional and/or transversal skills targeted by the subject for which the course description is prepared will be copied from the curriculum of the degree programme. For each competency, the complete entry, including the competency code, will be copied with the exact wording that appears in the curriculum, without any changes.

Professional competencies	
Competency code	**Competency**
PC16	Create software
PC27	Use software design patterns
PC14	Analyze software specifications
Transversal competencies	
Competency code	**Competency**
TC4	Solve problems
TC5	Think analytically

6.2. Learning outcomes relevant to the degree programme (as referred to in the curriculum)[2]

Learning outcomes targeted by the subject		
Competency code	**Knowledge and comprehension**	**Specific academic skills**
CP4 **CT2** **CP14**	*The student/graduates identifies, explains, and argues fundamental concepts of data structures, algorithms, and programming paradigms, as well as computer architecture*	*The student/graduates designs, develops, and demonstrates complex software solutions using efficient algorithms and diverse programming paradigms*
TC6	*The student/graduate identifies and describes the concepts studied in mathematics and computer science and correlates them with concepts from the English language*	*The student/graduate communicates fluently in English, both in written and oral form, conveying both scientific information and information from everyday life*

7. Subject-specific learning outcomes

Knowledge and comprehension
1. To understand the concepts of the object-oriented programming paradigm and to design object-oriented solutions of small/medium scale problems, using C++ and Qt.
2. To demonstrate the differences between traditional imperative design and object-oriented design.
3. To explain class structures as fundamental, modular building blocks.
4. To understand the role of inheritance, polymorphism, dynamic binding and generic structures in building reusable code.
Specific academic skills
1. To explain and to use defensive programming strategies, employing formal assertions and exception handling.
2. To design user- interfaces and write small/medium scale C++ programs using Qt.
3. To use classes written by other programmers and third-party libraries when constructing their systems.

8. Contents

If no competency is copied from either of the two categories, the row corresponding to that category is deleted from the table.

[2] The learning outcomes relevant for the degree programme and targeted by the subject for which the course description is prepared will be listed. The entries, copied without any changes from the Curriculum by subject type (Core Subject/Specialisation Subject/Complementary Subject), are listed under the corresponding competency.

8.1. Course	Teaching and learning methods	Remarks[3]
C/C++ introduction (basic elements of C/C++ programming language, data types, constant variables, scope and lifetime of the variables, statements, functions: declaration and definition, overloading functions).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Modular programming in C/C++ (functions, formal and actual parameters, pointers and memory management, the stack and the help, pointers to functions, header files, modular programming, libraries).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
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).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Inheritance and polymorphism (base and derived classes, Liskov substitution principle, method overriding, inheritance and polymorphism).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Polymorphism (static and dynamic binding, virtual methods, multiple inheritance, upcasting and downcasting, abstract classes, UML class diagrams and relations).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
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.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
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).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
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)	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Graphical User Interfaces (Qt Toolkit: installation, Qt modules and instruments, Qt GUI components, Layout management, design interfaces using Qt Designer).	• Interactive exposure • Explanation • Conversation • Examples	

[3] For example, organisational aspects, recommendations for students, specific aspects relating to the course/seminar, such as inviting experts in the field, etc.

	• Didactical demonstration	
Event driven programming I (callbacks, events, signals and slots in Qt).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Event driven programming II (Model View Controller, Models and Views in Qt, using predefined models, implementing custom models).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Design patterns I (creational, structural, behavioral patterns, examples, singleton, factory method, adapter pattern).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Design patterns II (façade pattern, observer pattern, strategy pattern, case study application and examples).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
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. S. Meyers. ***Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)***, Addison-Wesley, 2005.
4. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995.
5. B. Stroustrup. ***A Tour of C++,*** Addison-Wesley, 2013.
6. C++ reference (http://en.cppreference.com/w/).
7. Qt Documentation (http://doc.qt.io/qt-5/).
8. E. Gamma, R. Helm, R. Johnson, J. Vlissides. ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison-Wesley Longman Publishing, 1995.

8.2. Laboratory	**Teaching and learning methods**	**Remarks**
Environment setup (installing a C++ compiler and an IDE). C/C++ basics.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Introductory problems (in C).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Modular programming	• Interactive exposure • Explanation	

	• Conversation • Examples • Didactical demonstration	
Classes and objects in C++. Copy constructors, assignment operators, destructors.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Inheritance. Method overriding.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Inheritance and polymorphism. Virtual methods.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Laboratory test I	Evaluation	
STL containers, iterators and algorithms.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Streams, overloading the insertion and extraction operators, persistence.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Exception handling. Testing.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Qt Graphical User Interfaces I.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Qt Graphical User Interfaces II. Signals and slots in Qt.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Design patterns.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	

Laboratory test II	Laboratory test 2	

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 modernă în C++: Programare generică și 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. Bruce Eckel. ***Thinking in C++***, Prentice Hall, 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 Seminar	**Teaching and learning methods**	**Remarks**
Simple problems in C. Functions. Structures, enums and arrays.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	Throughout the seminar, the students will iteratively implement an application that involves converting text into Morse Code. In the first seminars they will write the classes to generate basic waves and save them in .csv files that will be converted to .wav files using a provided python script. Next, they will implement a layered architecture to store and play transmission messages.
Modular programming.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Classes. Operator overloading. User-defined objects as class data members. Files.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Inheritance. Polymorphism. Templates.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Exceptions. STL containers, iterators, algorithms.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Implementation based on UML diagrams. Design patterns (Command and Filter design patterns).	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
Graphical User Interfaces.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	

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 modernă în C++: Programare generică și 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. Bruce Eckel. ***Thinking in C++***, Prentice Hall, 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.

9. Evaluation

Type of activity	9.1 Evaluation criteria[4]	9.2 Evaluation methods[5]	9.3 Percentage in the final grade
9.4. Course	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct C++ programs.	Written examination (regular session).	60% of the final grade.
9.5. Seminar/ laboratory	Ability to design, implement, test and debug a C++ program (with a graphical user interface).	Practical evaluation. 7th week of the semester	20% of the final grade
		Practical evaluation. 14th week of the semester	20% of the final grade
9.6 Minimum standard for passing			

- 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 and for the written examination.
- The written exam is organised in two parts: a theoretical section and a practical section. Students must obtain a grade of at least 5 in both parts.
- Attendance is mandatory for 5 seminar sessions and 12 laboratory sessions.

10. SDG labels (Sustainable Development Goals)[6]

[4] The evaluation criteria must directly reflect the learning outcomes targeted at the level of the degree programme respectively at the level of the subject. More specifically, the learning outcomes set out in the expected learning outcomes are assessed.

[5] Both final evaluation methods and ongoing evaluation strategies should be established.

[6] Select a single label which, according to the *Implementation of SDG labels in the academic process*, best matches the subject. If the subject addresses sustainable development in a generic manner (i.e. by presenting/introducing the general framework of sustainable development, etc.), then the Sustainable Development generic label may be applied. If none of the labels describe the subject, select the last option: *"No label applies."*

	○	Sustainable Development Generic Label

								
○	○	○	○	○	○	○	○	x
								No label applies
○	○	○	○	○	○	○	○	○

Date of entry:
22.05.2026

Signature of course coordinator

Lect. PhD. Diana-Laura Borza

Signature of seminar coordinator

Lect. PhD. Diana-Laura Borza

Date of approval in the department:
...

Signature of the head of department

.....