SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş-Bolyai University of Cluj-Napoca**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Departament	**Departament of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Ciclul de studii	**Bachelor**
1.6 Study cycle / Qualification	**Computer Science - English**

2. Information regarding the discipline

2.1 Name of the discipline	**Object Oriented Programming**						
2.2 Course coordinator	**Assoc. Prof. PhD Bocicor Maria Iuliana**						
2.3 Seminar coordinator	**Assoc. Prof. PhD Bocicor Maria Iuliana**						
2.4 Year of study	**1**	2.5 Semester	**2**	2.6. Type of evaluation	**E**	2.7. Type of discipline	**Compulsory**

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

3.1 Hours per week	5	Of which: 3.2 course	2	3.3 seminar/laboratory	1sem 2 lab
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	14+ 28
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					24
Additional documentation (in libraries, on electronic platforms, field documentation)					15
Preparation for seminars/labs, homework, papers, portfolios and essays					19
Tutorship					9
Evaluations					13
Other activities:					

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 activities	• Laboratory with computers; C++ and programming language and Qt library

6. Specific competencies acquired

Professional competencies	• 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.
Transversal competencies	• 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)

7.1 General objective of the discipline	• To prepare an object-oriented design of small/medium scale problems and to learn the C++ programming language, as well as to create graphical user interfaces using Qt.
7.2 Specific objectives of the discipline	• To demonstrate the differences between traditional imperative design 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 write small/medium scale C++ programs using Qt. • To use classes written by other programmers when constructing their systems.

8. Content

8.1 Course	Teaching methods	Remarks
1. Basic elements in C • Basic elements of C/C++ language • Lexical elements. Operators. Conversions • Data types. Variables. Constants • Visibility scope and lifetime of the variables • C++ Statements	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	

• Function declaration and definition. Function overloading. Inline functions		
2. Modular programming in C/C++ • Functions. Parameters • Pointers and memory management • Function pointers • Header files. Libraries • Modular implementations of ADTs	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
3. Object oriented programming in C++ • Classes and objects • Defining classes • Object creation and destruction • Operator overloading • Static and friend elements	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
4. Templates and the Standard Template Library • Function templates • Class templates • Containers, iterators in STL • STL algorithms	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
5. Inheritance • Simple inheritance and derived classes • Special functions in classes and inheritance • Substitution principle • Method overriding • Multiple inheritance • UML class diagrams and relations	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
6. Polymorphism • Inheritance, polymorphism • Static and dynamic binding • Virtual methods • Upcasting and downcasting • Abstract classes	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
7. Streams and exception handling • Input/Output streams • Insertion and extraction operators • Formatting. Manipulators. Flags • Text files • Exception handling. Exception-safe code	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
8. Resource management and RAII • Resource Acquisition Is Initialization (RAII) • Smart pointers • RAII in STL. Smart pointers in STL	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
9. Graphical User Interfaces (GUI) • Qt Toolkit: installation, Qt modules and instruments • Qt GUI components • Layout management • Qt Designer	• Interactive exposure • Explanation • Conversation • Examples • Didactical	

	demonstration	
10. Event driven programming elements • Callbacks • Events. Signals and slots in Qt • GUI design	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
11. Event driven programming elements • Model View Controller pattern • Models and Views in Qt • Using predefined models. Implementing custom models • Case study: Gene manager application	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
12. Design patterns • Creational, structural, behavioural patterns • Examples	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
13. Design patterns • Adapter pattern • Observer pattern • Iterator pattern • Composite 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
1. Simple problems in C. Functions. Structures and vectors.	• Interactive exposure	The seminar is

2. Modular programming.	• Explanation • Conversation • Examples • Didactical demonstration	structured as a 2 hour class, every 2 weeks.
3. Classes. Operator overloading. User defined objects as class data members. Templates (dynamic vector).		
4. Inheritance, polymorphism.		
5. Files, exceptions. STL containers, iterators, algorithms.		
6. Graphical User Interfaces		
7. Complex problems. Implementation based on UML diagrams. Design patterns.		

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/).

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. Setting up a C++ compiler (MSVC/MinGW) and an IDE (Visual Studio). C/C++ general aspects.	• Explanation • Conversation	• The laboratory is structured as weekly 2 hour classes. • Laboratory assignments are due 1 week after assignment.
2. Simple problems (in C).		
3. Feature-driven software development process. Layered architecture. Test driven development. Modular programming. (I)		
4. Feature-driven software development process. Layered architecture. Test driven development. Modular programming. (II)		
5. Object oriented programming in C++. (I)		
6. Object oriented programming in C++. (II)		
7. Laboratory test.		
8. Inheritance and polymorphism.		
9. Text Files, exceptions. STL containers, iterators and algorithms.		
10. Laboratory test.		
11. Qt Graphical User Interfaces. (I)		
12. Qt Graphical User Interfaces. (II)		
13. Laboratory test.		
14. Assignment delivery time.		

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.

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 follows 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 Lecture	The correctness and completeness of the accumulated knowledge and the capacity to design and implement correct C++ programs.	Written examination (regular session)	**30%**
10.5 Seminar/ Laboratory	Be able to design, test and debug a C++ program with a graphical user interface.	Practical examination (regular session)	**30%**
	Correctness of delivered laboratory assignments and laboratory tests.	Program and documentation portfolio. Observation during the semester. Laboratory tests.	**40%**
10.6 Minimum performance standards			

- Each student has to 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 knowledge in a coherent form, that they have the ability to establish certain connections and to use the knowledge in solving different problems in object oriented programming in C++.
- Attendance is compulsory for seminar (minimum 5) and laboratory (minimum 12) activities.
- Successfully passing of the examination is conditioned by a minimum grade of 5 for each of the following: laboratory activity, practical test and written examination.

Date	Signature of course coordinator	Signature of seminar coordinator
27.04.2020	Assoc. Prof. PhD. Bocicor Maria Iuliana	Assoc. Prof. PhD. Bocicor Maria Iuliana

Date of approval	Signature of the head of department
	Prof. PhD. Anca Andreica