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 Department	**Departament of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science**

2. Information regarding the discipline

2.1 Name of the discipline	**Object Oriented Programming**						
2.2 Course coordinator	**Lect. PhD Czibula Istvan Gergely**						
2.3 Seminar coordinator	**Lect. PhD Czibula Gabriela Gergely**						
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	1 sem 2 lab
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	42

Time allotment:	hours
Learning using manual, course support, bibliography, course notes	30
Additional documentation (in libraries, on electronic platforms, field documentation)	19
Preparation for seminars/labs, homework, papers, portfolios and essays	24
Tutorship	14
Evaluations	18
Other activities:	-

3.7 Total individual study hours	105
3.8 Total hours per semester	175
3.9 Number of ECTS credits	7

4. Prerequisites (if necessary)

4.1. curriculum	Fundamentals of Programming, Data Structures
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 QT programming language environment

6. Specific competencies acquired

Professional competencies	• Understanding the concepts of object oriented programming. • Understanding the role of inheritance, polymorphism, dynamic binding and generic structures in building reusable code. • Good programming skills in C++ and QT.
Transversal competencies	• The ability to apply the acquired concepts, principles and techniques in solving real world problems. • Responsible execution of lab assignments. • Application of efficient and rigorous working rules. • Manifest responsible attitudes toward the scientific and didactic fields. • Respecting the professional and ethical principles.

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 C++ and QT.
7.2 Specific objective 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. The Object Oriented Programming Paradigm. • Basic elements of C++ language. • Lexical elements. Operators. Conversions. • Data types. Variables. Constants. • Visibility scope and lifetime of the variables. Namespaces. • C++ Statements. • Function declaration and definition. Function overloading. Inline function.	• Interactive exposure • Explanation • Conversation • Examples • Didactical demonstration	
2. Modular programming in C++. • Functions. Parameters. • Header files. Libraries. • Modular implementations of ADTS. • Using the void pointer to achieve genericity.	• Interactive exposure • Explanation • Conversation • Examples • Didactical	

	demonstration	
3. Derived data types and user data types, dynamic allocation in C++. • Data types: array and struct. • Data types: pointer and reference. • Memory allocation and deallocation. • Pointers to functions and pointers void.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
4. Object oriented programming in C++. • Classes and objects. • Members of a class. Access modifiers. • Constructors / destructors • UML diagrams for classes (members, accessibility).	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
5. Inheritance • Simple inheritance. Derived classes. • Substitution principle. • Method overriding. • Multiple inheritance. • Specialization/generalization relation - UML representation.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
6. Input/output operation. • I/O streams. I/O Hierarchies of classes. • Format. Manipulators. • Text files.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
7. QT Toolkit. • QT tools and modules. • QT Installation. • Examples	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
8. QT • Signals and slots. • QWidget. • Examples	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
9. Working with QT Designer in Eclipse (1) • Design of GUI • Master detail – Product. Case study	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
10. Working with QT Designer in Eclipse (2) • Master detail – Product. Case study • MVC pattern	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
11. Design patterns • Creational, structural, behavioral design patterns. • Examples. STL library. • Container classes.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
12. STL library • STL iterators.	• Interactive exposure • Explanation	

• STL allgorithms	• Conversation • Didactical demonstration	
13. POS (Point Of Sale) application • Façade, Strategy design patterns • Composite design pattern	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
14. Revision	• Interactive exposure • Conversation	

Bibliography

1. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel, Thinking in C++, www.bruceeckel.com
3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002
4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra,Cluj-Napoca 1996.

8.2 Seminar	Teaching methods	Remarks
		The seminar is structured as 2 hours classes every two week
1. Simple problems in C++. Functions. Function parameters. Variables (local and global) and their visibility. Vectors (uni and multi dimensional) and structures.	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
2. ADT Container with generic elements (void*): visible representation and hidden representation.	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
3. Classes. Simple classes. Operator overloading. Classes with objects as data members .	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
4. Classes of type dynamic list and iterators. Inheritance.	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
5. Abstract classes and interfaces. Polymorphism	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
6. Classes: template and exceptions	• Interactive exposure • Explanation • Conversation	

	• Didactical demonstation	
7. Complex problems implementing by following the UML diagram. Design patterns. Preparation for the written exam.	• Interactive exposure • Explanation • Conversation • Didactical demonstation	
8.3 Laboratory	Teaching methods	Remarks
		• The lab is structured as 2 hours classes every week. • The lab documents are due one week after the lab theme has been given and the lab programs are due two weeks later.
1. Installation of MinGW and Eclipse CDT Specification, design and implementation of simple problems in C/C++. General aspects of C/C++ language.	• Lab assignment • Explanation • Conversation	
2. Modular programming in C++	• Lab assignment • Explanation • Conversation	
3. Feature driven software development process	• Lab assignment • Explanation • Conversation	
4. Feature driven software development process	• Lab assignment • Explanation • Conversation	
5. Feature driven software development process	•	
6. Layered architecture	• Lab assignment • Explanation • Conversation	
7. Layered architecture	• Lab assignment • Explanation • Conversation	
8. Layered architecture	• Lab assignment • Explanation • Conversation	
9. Text files	• Lab assignment • Explanation • Conversation	
10. GUI using QT	• Lab assignment • Explanation • Conversation	
11. Repository.	• Lab assignment • Explanation • Conversation	
12. STL containers, iterators and algorithms	• Lab assignment	

	- Explanation - Conversation	
13. Lab delivery time (see remark above)	- Lab assignment - Explanation - Conversation	
14. Lab delivery time (see remark above)	- Lab assignment - Explanation - Conversation	

Bibliography

1. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel, Thinking in C++, www.bruceeckel.com
3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002
4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra,Cluj-Napoca 1996.

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 IEEE and 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 the software companies as important for average 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 exam (in the regular session)	40%
10.5 Seminar/Lab activities	- Be able to design, test and debug a C++ program using QT	Practical evaluation (in the regular session)	30%
	- Correctness of C++ programs and lab documentations	-documentation -portofolio -continuous observations	30%
10.6 Minimum performance standards			

- Each student has to prove that (s)he acquired an acceptable level of knowledge and understanding of the, that (s)he is capable of stating these knowledge in a coherent form, that (s)he has the ability to establish certain connections and to use the knowledge in solving different problems in C++ programming language.
- Successful passing of the exam is conditioned by the final grade that has to be at least 5.

Date	Signature of course coordinator	Signature of seminar coordinator
30.04.2013	Lect. dr. Istvan Gergely Czibula	Lect. dr. Istvan Gergely Czibula

Date of approval	Signature of the head of department
	Prof. dr. Bazil Pârv