SYLLABUS

Data Structures

1. Information on academic programme

1.1. University	„1 Decembrie 1918” from Alba Iulia
1.2. Faculty	Faculty Of Exact Sciences and Engineering
1.3. Department	Informatics, Mathematics and Electronics Department
1.4. Field of Study	Computer Science
1.5. Cycle of Study	Undergraduate
1.6. Academic programme / Qualification	Computer Science / 251201, 251203, 251204

2. Information of Course Matter

2.1. **Course**		*Data Structures*			2.2. **Code**		CSE109
2.3. **Course Leader**			Rotar Corina				
2.4. **Seminar Tutor**			Cristea Daniela				
2.5. Academic Year	**I**	2.6. Semester	**II**	2.7. Type of Evaluation (E – final exam/ CE - colloquy examination / CA -continuous assessment)	**E**	2.8. Type of course (**C**– Compulsory, **Op** – optional, **F** - Facultative)	**C**

3. Course Structure (Weekly number of hours)

3.1. Weekly number of hours	**4**	3.2. course	***2***	3.3. seminar, laboratory	***3***
3.4. Total number of hours in the curriculum	**56**	3.5. course	***28***	3.6. seminar, laboratory	***42***
Allocation of time:					Hours
Individual study of readers					***25***
Documentation (library)					***20***
Home assignments, Essays, Portfolios					***50***
Tutorials					**-**
Assessment (examinations)					***10***
Other activities…….					**-**

3.7 Total number of hours for individual study	**105**
3.9 Total number of hours per semester	**175**
3.10 umber of ECTS	**7**

4. Prerequisites (*where applicable*)

4.1. curriculum-based	Fundamentals of programming
4.2. competence-based	C1.1 The appropriate description of programming paradigms and of specific language mechanisms, as well as the identification of differences between semantic and syntactic aspects. C1.3 The development of correct source codes and the testing of various components in a known programming language, given a set of design specifications

5. **Requisites** (*where applicable*)

5.1. course-related	*Room equipped with video projector / board*
5.2. seminar/laboratory-based	*Laboratory – computer, Software: Visual Studio 2010, BorlandC, Internet access.*

6. Specific competences to be aquired (chosen by the course leader from the programme general competences grid)

Professional competences	***C1 Programming in high-level languages*** C1.1 The appropriate description of programming paradigms and of specific language mechanisms, as well as the identification of differences between semantic and syntactic aspects. C1.2 The explaining of existing software applications using different abstraction layers (architecture, packages, classes, methods), correctly using base knowledge. C1.3 The development of correct source codes and the testing of various components in a known programming language, given a set of design specifications. C1.4 The testing of various applications given specific testing plans C1.5 Developing program units and their documentation.
Transversal competences	Not applicable

7. Course objectives (as per the programme specific competences grid)

7.1 General objectives of the course	Develop students' ability to design software that is dedicated to solving medium complexity problems. Deepening the concept of data structure and gaining the skills to design abstract data types and associated libraries. Creating a rigorous and efficient programming style
7.2 Specific objectives of the course	Developing students' ability to effectively manage information by using abstract data types and rigorously designing the algorithms to process the data. Drawing a coherent documentation on the applications of average complexity.

8. Course contents

8.1 Course (learning units)	**Teaching methods**	**Remarks**
1. Introduction. Programming paradigms	***Lecture, conversation, exemplification***	**2**
2. Data structures. Abstract data type (ADT). Examples: Rational ADT, Compex ADT- 2 sessions	***Lecture, conversation, exemplification***	**2**
3. Simple linked lists, circulars, stack, queue. List ADT.	***Lecture, conversation, exemplification***	**2**
4. Double Linked lists	***Lecture, conversation, exemplification***	**2**
5. ADT Trees	***Lecture, conversation, exemplification***	**2**
6. ADT tables	***Lecture, conversation, exemplification***	**2**
7. TAD Graphs. Algorithms on graphs.	***Lecture, conversation, exemplification***	**2**
8. Programming methods. Divide et Impera technique.	***Lecture, conversation, exemplification***	**2**
9. Greedy method.	***Lecture, conversation, exemplification***	**2**
10. Branch and Bound method.	***Lecture, conversation, exemplification***	**2**
11. Backtracking method. - 2 sessions	***Lecture, conversation, exemplification***	**2**
12. Dynamic programming method.	***Lecture, conversation, exemplification***	**2**
Seminars-laboratories	**Teaching methods**	**Remarks**
1. Review programming paradigms. Moderately	***Project-work, computer-based***	**3**

complex problems with different data structures used	***activities, laboratory activities***	
2. Data structures. ADT Compex implementation.	***laboratory activities***	**3**
3. Simple linked lists, circulars lists, stacks, queues. ADT List.	***laboratory activities***	**3**
4. Double linked list.	***laboratory activities***	**3**
5. Trees.	***laboratory activities***	**3**
6. Binary search tree. Operations on trees.	***laboratory activities***	**3**
7. ADT tables	***laboratory activities***	**3**
8. ADT graphs. Graphs' representation	***laboratory activities***	**3**
9. Algorithms on graphs.	***laboratory activities***	**3**
10. Programming methods. Divide et Impera techniques.	***laboratory activities***	**3**
11. Greedy method-specific issues	***laboratory activities***	**3**
12. Branch and Bound method-specific issues	***laboratory activities***	**3**
13. Backtracking method-specific issues	***laboratory activities***	**3**
14. Dynamic programming method-specific issues	***laboratory activities***	**3**

References

1. Rotar C., Data structers and algorithms, Ed. Didactica, Alba Iulia, 2008.
2. Bruce Eckel, Thinking in C++, manual online.
3. Bjarne Stroustrup, The C++ Programming Language, Addison Wesley, 1997.
4. H. Schildt: C++ manual complet, electronic book.
5. Peter Muller: Introduction to Object-Oriented Programming Using C++ , electronic book.

1. Corroboration of course contents with the expectations of the epistemic community's significant representatives, professional associations and employers in the field of the academic programme

Not applicable. *Algorithms and Data Structure* is a fundamental subject in the domain which is required in the curricula of Computer Science specialization. Course content is designed for training the algorithmic thinking of the students.

2. Assessment

Activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Percentage of final grade
10.4 Course	*Final evaluation*	*Written paper*	60%
	-	-	-
10.5 Seminar/laboratory	*Continuous assessment*	*Laboratory activities portfolio*	40%
	-		-
10.6 Minimum performance standard:			
Implementation and documentation of the software units in high-level programming languages and efficiently used programming environments			

Submission date　　　　Course leader signature　　　　Seminar tutor signature

Date of approval by Department members　　　　Department director signature