Syllabus

1. Programme information

1.1. Institution	THE BUCHAREST UNIVERSITY OF ECONOMIC STUDIES
1.2. Faculty	Economic Cybernetics, Statistics and Informatics
1.3. Departments	Department of Economic Informatics and Cybernetics
1.4. Field of study	Economic Cybernetics, Statistics and Informatics
1.5. Cycle of studies	Licence
1.6. Education type	Full-time
1.7. Study programme	Economic informatics (in english)
1.8. Language of study	English
1.9. Academic year	2020-2021

2. Information on the discipline

2.1. Name	**Data Structures**								
2.2. Code	**20.0233IF2.2-0004**								
2.3. Year of study	**2**	2.4. Semester	**2**	2.5. Type of assessment	**Exam**	2.6. Status of the discipline	**O**	2.7. Number of ECTS credits	**4**

2.8. Leaders	C(C)	**conf.univ.dr. DOINEA Laurenţiu-Mihai**	mihai.doinea@ie.ase.ro
	S(S)	**conf.univ.dr. DOINEA Laurenţiu-Mihai**	mihai.doinea@ie.ase.ro
	S(S)	**lect.univ.dr. ZAMFIROIU I IONUŢ ALIN**	alin.zamfiroiu@csie.ase.ro

3. Estimated Total Time

3.1. Number of weeks	14.00		
3.2. Number of hours per week	4.00	of which	
		C(C)	2.00
		S(S)	2.00
3.3. Total hours from curriculum	56.00	of which	
		C(C)	28.00
		S(S)	28.00
3.4. Total hours of study per semester (ECTS*25)	100.00		
3.5. Total hours of individual study	44.00		
Distribution of time for individual study			
Study by the textbook, lecture notes, bibliography and student's own notes	24.00		
Additional documentation in the library, on specialized online platforms and in the field	5.00		
Preparation of seminars, labs, assignments, portfolios and essays	9.00		
Tutorials	1.00		
Examinations	3.00		
Other activities	2.00		

4. Prerequisites

4.1. of curriculum	IT&C Fundamentals; Computer Programming Fundamentals Programming Techniques; Object-Oriented Programming
4.2. of competences	Knowledge of the ways in which data are organized in computer memory, models and requirements of data usage in the source program, the use and allocation of memory areas; Knowledge of the syntax elements in C++ programming language and how to build, compile, execute and debug the source program.

5. Conditions

for the C(C)	Lectures held in rooms with multimedia teaching equipment (video-projector)
for the S(S)	Rooms with computers installed with C + + programming environment.

6. Acquired specific competences

PREFESSIONAL	C4	Developing software components by using data structures, algorithms, techniques and modern programming languages.

7. Objectives of the discipline

7.1. General objective	Initiation in the definition and proper use of data structures in developing software components.
7.2. Specific objectives	Assimilation of the techniques and methods for design and implementation of data structures in the development of software components; Selecting the appropriate data structures for each software component based on practical criteria; Using data structures in complex software projects focused on economic issues.

8. Contents

8.1. C(C)		Teaching/Work methods	Recommendations for students
1	Review. Standard and software developer-defined data types. Pointers. Models and requirements for defining, initializing, using and readability of data in the source program. Stack memory and Heap memory. Quality indicators for using the memory	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
2	Noncontiguous dynamic data structures: simple linked list and double linked list – definition, allocation and use. Stack and queue – definition, allocation and use.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
3	Sparse matrix: definition, allocation and use. Heterogeneous and contiguous data structures. Implementation of the sparse matrices by arrays and heterogeneous data structures	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
4	The graph data structure: caracteristics, definition, allocation and use. Traversal algorithms of the graph. Connectivity and connectivity paths algorithms.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	

5	Tree data structures: arbitrary tree structures and binary trees – definition, allocation and use. Structure trees: characteristics, implementation and operations. Evaluation of a mathematical expression using structure trees.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
6	Tree data structures: binary search tree – definition, allocation and use.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
7	Balanced trees and self-balancing search trees: balanced binary trees, AVL trees, Red-Black trees – definition, caracteristics and operations.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
8	The B-tree data structure: definition, properties, allocation, algorithms and implementation of the fundamental operations (insertion and deletion).	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
9	Hash tables: caracteristics, hash functions, operations, collision avoidance mechanisms.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
10	Heap data structure – definition, allocation and use. Priority queues.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	
11	Data compression: caracteristics, classification and data compression algorithms. Conversions of data structures: caracteristics of conversion process, conversion types, productive elements at process level, respectively at application level.	Lectures on PowerPoint support, examples of implementation in development environment, interaction with students.	

Bibliography

- Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori), Structuri de date, ASE, Bucureşti, 2008, România
- Ion Smeureanu, Programarea in limbajul C/C++, CISON, Bucureşti, 2001, România
- Saumeyendra Sengupta, Carl Phillip Korobkin, C++ Object Oriented Data Structures, Springer Verlag, New York, 1994, Statele Unite ale Americii
- William Ford, William Topp, Data Structures with C++, Prentice Hall, New Jersey, 1996, Statele Unite ale Americii
- E. Demaine, Advanced Data Structures, 2003, http://courses.csail.mit.edu/ 6.897/spring03/scribe_notes, Statele Unite ale Americii
- http://www.acs.ase.ro, România
- http://www.dice.ase.ro, România

8.2. S(S)		Teaching/Work methods	Recommendations for students
1	Exemples for defining, initializing and referring the standard and developer-defined data. The access and referring the data by pointers. Addressing the stack and heap memories and associations between these. Determination of quality indicators for allocation and using the memory areas.	Examples of implementation in programming environment.	
2	Exemples on simple and double lists containing values of standard and programmer-defined data types. Implementation and using the list allocated in dynamic memory completely. Use of debugging tools from development environment approaching the management of memory areas allocated by programmer. Practical exemples for implementation and use of the stacks and queues using noncontiguous dynamic data structures. Application: implementation of the mathematical expression in the postfix form.	Examples of implementation in programming environment.	
3	Practical exemples for spare matrix implementation and operation on sparse matrices.	Examples of implementation in programming environment.	
4	Exemples implementing the graph structure on adjacency matrix and adjacency list. Implementation of the traversal operations of the graph.	Examples of implementation in programming environment.	
5	Practical examples for connectivity algorithm implementation.	Examples of implementation in programming environment.	
6	Practical exemples implementing the tree structures. Implementation of the specific operations working with tree structures.	Examples of implementation in programming environment.	
7	Practical exemples implementing the tree structures. Implementation of the specific operations working with tree structures.	Examples of implementation in programming environment.	
8	Practical examples for implementation of operations for balancing the binary and AVL trees.	Examples of implementation in programming environment.	
9	Implementation and operations on structure trees.	Examples of implementation in programming environment.	
10	Practical exemples implementing the B-tree structure, implementation of the insertion and deletion algorithms for a key in B-tree structure.	Examples of implementation in programming environment.	
11	Implementations of hash tables and heap data structure. Implementation of the priority queues.	Examples of implementation in programming environment.	

Bibliography

- Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori), Structuri de date, ASE, Bucureşti, 2008, România
- Ion Ivan, Cristian Ionita, Cătălin Boja, Marius Popa, Adrian Pocovnicu, Daniel Milodin, Practica dezvoltării software orientată pe structuri de date, ASE, Bucureşti, 2005, România
- Ion Smeureanu, Programarea in limbajul C/C++, CISON, Bucureşti, 2001, România
- Bjarne Strastroup, The C++ Programming Language, Addison-Wesley, http://www.research.att.com/~bs/3rd.html, Statele Unite ale Americii
- http://www.acs.ase.ro
- http://www.dice.ase.ro

9. Corroboration of the contents of the discipline with the expectations of the representatives of the epistemic community, of the professional associations and representative employers in the field associated with the programme

In the software industry, products have a very high level of complexity determined by the number of software components, linkages among components, implemented algorithms, deployed functionality and nature of the problem that the software addresses.
In the software engineering knowledge of data structures, their implementation and operations classes that are implemented based on their complexity is essential in improving the software complexity and its development process. These elements are key factors in achieving high quality software products made with minimal cost and deadlines by software producers.

10. Assessment

Type of activity	Assessment criteria	Assessment methods	Percentage in the final grade
10.1. S(S)	Assessment of implemented data structures and responses to the lecturer's questions by the students.	Online computer tests and challenges on data structure implementation topics with synchronous and asynchronous assessment.	40.00
10.2. Final assessment	Assessment of implemented data structures and responses to the lecturer's questions by the students.	Online practical test written by using a computer.	60.00
10.3. Modality of grading	Whole notes 1-10		
10.4. Minimum standard of performance	Exam evaluation scale: Mark 5: Compiling, running and debugging the application in IDE, compliance with the application logic. Application modularity. Mark 6: Definition and adequate use of the data structures required to be implemented in the application. Mark 7: Adequate allocation of the data structures in memory. Dezallocation of the memory areas used to implementing the data structures. Mark 8: Implemention the operations required for the allocated data structures. Mark 9: Defining the test data sets and validation of the results according to implemented logical flow. Mark 10: Optimal use of the memory areas and the management of the memory areas. Observations: Granting grade 5 or greater than 5 is conditioned by the accumulation of at least 50% of the evaluation points awarded for the final assessment (by exam). Homework and exam projects will be verified using dedicated software for plagiarism detection or through search engines. The existence of a level of similarity of over 49% will lead to the failure to pass the exam and to the initiation of the necessary measures provided by the university regulations.		

Date of listing,
10/25/2021

Signature of the discipline leaders,

Date of approval in the department

Signature of the Department Director,