

Contributions in the Field of Generative Models Applied in eLearning

A Thesis Submitted for obtaining
the Scientific Title of PhD in Engineering
from
Politehnica University Timișoara
In the Field of Computers and Information Technology
by

Felicia Mirabela DUME (COSTEA)

PhD Committee Chair: Conf. dr. ing. Dan Pescaru
Sl. dr. ing. Răzvan Bogdan
Prof. dr. ing. Carmen Holotescu
PhD Supervisor: Prof. univ. dr. ing. Vladimir Ioan Crețu
Scientific Reviewers:

Date of the PhD Thesis Defense:

The PhD thesis series of UPT are:

- | | |
|--|---|
| 1. Automation | 11. Science and Material Engineering |
| 2. Chemistry | 12. Systems Engineering |
| 3. Energetics | 13. Energy Engineering |
| 4. Chemical Engineering | 14. Computers and Information Technology |
| 5. Civil Engineering | 15. Materials Engineering |
| 6. Electrical Engineering | 16. Engineering and Management |
| 7. Electronic Engineering and Telecommunications | 17. Architecture |
| 8. Industrial Engineering | 18. Civil Engineering and Installations |
| 9. Mechanical Engineering | 19. Electronics, Telecommunications
and Information Technologies |
| 10. Computer Science and
Information Technology | |

Politehnica University Timișoara, Romania, initiated the above series to disseminate the expertise, knowledge and results of the research carried out within the doctoral school of the university. According to the Decision of the Executive Office of the University Senate No. 14/14.07.2006, the series includes the doctoral theses defended in the university since October 1, 2006.

Copyright © Editura Politehnica – Timișoara, Romania, 2021

This publication is subject to copyright law. The multiplication of this publication, in whole or in part, the translation, printing, reuse of illustrations, exhibit, broadcasting, reproduction on microfilm or any other form is allowed only in compliance with the provisions of the Romanian Copyright Law in force and permission for use obtained in writing from the Politehnica University Timișoara, Romania. The violations of these rights are under the penalties of the Romanian Copyright Law.

România, 300159 Timișoara, Bd. Republicii 9,
Tel./fax 0256 403823
e-mail: editura@edipol.upt.ro

Foreword

This thesis has been elaborated during my activity in the Department of Computers and Information Technology of the Politehnica University Timișoara, Romania.

The scientific substantiation and elaboration of this doctoral thesis would have been impossible without the help, support, and guidance of special people who, through a high degree of professionalism and dedication, contributed to my training as a researcher, instilling the courage to go further.

I would like to thank my PhD supervisor, prof. univ. dr. ing. Vladimir Ioan Crețu, for his permanent guidance, support, trust, and encouragement during the period of doctoral preparation and thesis elaboration. I thank the professor for thus contributing to my professional training as a person.

I would also like to express my gratitude to the members of the evaluation committee for their advice and suggestions.

I would like to thank conf. dr. ing. Ciprian-Bogdan Chirilă for the special collaboration, support, and for the opportunity for professional development that he offered me.

At the same time, I would like to express my special thanks to my family for their encouragement and dedicated support, expressed during the doctoral research process.

Timișoara, June 2021

Felicia Mirabela Dume (Costea)

Recipients of dedication.

DUME (COSTEA), Felicia Mirabela

Contributions in the Field of Generative Models Applied in E-Learning

PhD theses of UPT, Series X, No. YY, Editura Politehnica, 200Z, 265 pages, 221 figures, 5 tables.

ISSN:

ISBN:

Keywords

e-learning, auto-generative learning objects,

Abstract

The integration of ICT in the training process has become important in the evolution of education. Learning objects have emerged as a new way of organizing learning content. Improving the use of learning objects is an important direction in the field of research, several models being created.

We propose an approach based on auto-generative learning objects to facilitate learning and assessment for the STEM disciplines. An auto-generative learning object is a pedagogical model developed by the tutor to meet a specific learning objective. It offers different exercises for each instance based on randomly instantiated variables. The answers given by the students are evaluated automatically.

We have modeled concepts from several STEM disciplines. After applying them to the class we found that they are effective and appreciated by students.

CONTENTS

List of tables	1
List of figures	1
1. Introduction	8
2. State of the Art in Learning Objects.....	14
2.1. Learning Objects Definitions	14
2.1.1. IEEE Learning Object Metadata Standard.....	15
2.1.2. Sharable Content Object Reference Model	18
2.1.3. Cisco Learning Object Model	20
2.1.4. Dublin Core Standard.....	23
2.1.5. The Learnativity Content Model	24
2.1.6. NETg Learning Object Model	25
2.1.7. Abstract Learning Object Model	26
2.1.8. H5P learning objects.....	28
2.2. Learning Objects in Learning Management Systems.....	30
2.2.1. Learning Objects in Moodle.....	30
2.2.2. Learning Objects in ILIAS	34
2.2.3. Learning Objects in Blackboard Learn.....	37
2.3. Generative Learning Objects.....	39
2.3.1. GLOs Coined by Tom Boyle.....	39
2.3.2. GLOs in the Works of Damasevicius and Stuikys.....	42
2.3.3. Moodle Coordinate Questions	43
2.3.4. Moodle Calculated Questions.....	45
2.4. Learning Objects in Other Works	47
2.4.1. Learning Objects Sequencing Papers.....	47
2.4.2. Learning Objects Improved by Object-Oriented Design.....	48
2.4.3. Learning Objects in Cloud Frameworks	49
2.4.4. Learning Objects Records - Experience API.....	51
2.5. Statistical concepts used in the assessment of the proposed e-learning approach.....	56
2.6. Comparison of Learning Objects.....	57
2.7. Summary	60
3. The Auto-generative Learning Object Model.....	62
3.1. AGLO creation by StepWay Abstractization and generalization	64
3.1.1. The abstraction algorithm.....	64
3.1.2. The abstraction process applied on an example	65
3.1.3. Abstractions for different concepts from STEM disciplines	67
3.2. Mathematical Definition and Structure	73
3.3. Semantics	76
3.4. Summary	78
4. Models for Middle School Arithmetic	79
4.1. Fractions AGLOs.....	79
4.1.1. Fractions Classification AGLO	79
4.1.2. Common Divisor AGLO	81
4.1.3. Amplification and Simplification of a Fraction AGLO.....	82
4.1.4. Comparing Fractions AGLOs.....	85
4.1.5. Fraction Addition AGLOs	87

4.1.6.	Fraction Subtraction AGLO	92
4.1.7.	Fraction Multiplication AGLO	93
4.1.8.	Fraction Power AGLO	95
4.1.9.	Fraction Division AGLO	97
4.1.10.	Finding a Percentage of a Whole AGLO	100
4.1.11.	Transforming a Decimal Fraction AGLO.....	101
4.1.12.	Transforming a Periodic Decimal Fraction AGLO.....	102
4.1.13.	Transforming a Mixed Periodic Decimal Fraction AGLO.....	104
4.1.14.	Introduction of the Whole into a Fraction AGLO	104
4.1.15.	Extracting the Whole from a Fraction AGLO	106
4.2.	AGLOs for intervals, equations with the module and inequalities	107
4.2.1.	Intervals AGLO.....	107
4.2.2.	Equations involving absolute values AGLO	110
4.2.3.	Inequations AGLO.....	112
4.3.	AGLOs for algebraic calculation formulas	117
4.3.1.	Application of abbreviated calculation formulas AGLO.....	118
4.3.2.	Decomposition into factors AGLO	119
4.4.	Summary	120
5.	AGLOs for Information Technology Disciplines	121
5.1.	Models for Data Structures and Algorithms Discipline.....	121
5.1.1.	Linear search AGLOs	121
5.1.2.	Linear search with sentinel AGLOs	131
5.1.3.	Binary search AGLOs.....	135
5.1.4.	Search by interpolation AGLOs	142
5.1.5.	Insertion sort AGLOs.....	148
5.1.6.	Selection sort AGLOs	153
5.1.7.	Bubble sort AGLOs.....	157
5.1.8.	Shell sort AGLOs	159
5.1.9.	Quicksort AGLOs	162
5.1.10.	Linked Lists AGLOs	165
5.1.11.	Double linked list AGLOs.....	172
5.2.	Models for Algorithm Analysis and Design Discipline	176
5.2.1.	AGLOs regarding the notion of tree.....	176
5.2.2.	AGLOs regarding the notion of graph	190
5.3.	Models for Operating Systems Discipline	199
5.3.1.	Commands for directories AGLOs.....	199
5.3.2.	Commands for files AGLOs.....	205
5.3.3.	Commands for processes AGLOs	213
5.3.4.	Commands for disk partitioning AGLOs.....	217
5.3.5.	Package management AGLOs.....	219
5.3.6.	Administrative commands AGLOs	221
5.3.7.	Basic networking commands AGLOs.....	226
5.4.	Summary	231
6.	Prototype Implementation.....	234
6.1.	The DSEL platform	234
6.2.	The domain-specific JavaScript Libraries	237
6.2.1.	The JavaScript Library for Fractions Applications	237

6.2.2.	The JavaScript Library for Intervals Applications.....	238
6.2.3.	The JavaScript Library for Data Structures and Algorithms.....	239
6.2.4.	The JavaScript Library for Design and Analysis of Algorithms	242
6.2.5.	The JavaScript Library for Operating System.....	243
6.3.	Summary	244
7.	Validation of Auto-generative Learning Objects.....	245
7.1.	Case Study for Fractions AGLOs	245
7.2.	Case Study for Intervals, Equations and Inequations AGLOs	248
7.3.	Summary	251
8.	Conclusions and Perspectives	253
8.1.	Conclusions	253
8.2.	Meeting the Objectives.....	255
8.3.	Future Work	256
	Bibliography	258

LIST OF TABLES

Table 2.1. LO comparison	58
Table 2. 2. Moodle plugins and AGLO comparison	59
Table 3. 1. The thesis objective associated with the thesis chapters	63
Table 6. 1. Distribution of AGLOs on topics for DSA discipline	232
Table 6.2. Distribution of AGLOs on topics for OS discipline	233
Table 8.1. Grades obtained in the two types of assessments	245

LIST OF FIGURES

Fig.1.1. The LOs conceptual hierarchy	9
Fig.1.2. The AGLO Approach.....	10
Fig.1.3. An instantiation of the addition of a node in an ordered linked list	11
Fig.2.1. The hierarchy of elements in the LOM data model.....	16
Fig.2.2. An example of a SCORM course	19
Fig.2.3. How LMS returns student status.....	20
Fig.2.4. The CISCO Learning Object structure.....	21
Fig.2.5. Course Hierarchy in CISCO Learning Object Model.....	21
Fig.2. 6. Dublin Core metadata encoding examples	24
Fig.2.7. The five-level Learnativity content hierarchy.....	25
Fig.2.8. The architecture of the NETg Model	26
Fig.2.9. The ALOCOM model architecture	28
Fig.2.10. An interactive video made in H5P	29
Fig.2.11. A blank Moodle course page using the Boost theme	31
Fig.2.12. Resources provided by Moodle	32
Fig.2.13. The activities offered by Moodle	33
Fig.2.14. LOs in ILIAS	35
Fig.2.15. An e-campus ILIAS page	37
Fig.2.16. A course on Blackboard Learn	38
Fig.2.17. Schematic layout of the format for LO realization	39
Fig.2.18. Representation of functional choices in GLO-Maker	40
Fig.2.19. Examples of the 'surface' structure 'pages' of GLOs as seen by learners .	41
Fig.2.20. The structure of the GLO model.....	42
Fig.2.21. Example of LO: Bubble sort; C++; on Mobile (a fragment)	43
Fig.2.22. Example of Moodle Coordinate Question	44
Fig.2.23. Moodle Calculated Question example	46
Fig.2.24. The LO sequencing process.....	47
Fig.2.25. OOGLOM as a base for IS/OS Models	48
Fig.2.26. Example of LO in CLAVIRE	51
Fig.2.27. A Statement that voids a previous Statement	53
Fig.2.28. Example of a simple statement	53
Fig.2.29. Statement reference example	54
Fig.2.30. Example of the "context" field	55
Fig.4.1. Sketch for making an AGLO.....	66
Fig.4.2. Sketch for making the fractions addition AGLOs	67
Fig.4.3. Sketch for making the creation of a hierarchy of directories AGLOs	73

Fig.4.4 The Grammar	75
Fig.4.5. AGLO Name and Scenario	77
Fig.4.6. AGLO Theory and Question.....	77
Fig.4.7. AGLO Answer and Feedback	78
Fig.5.1. The Scenario for the fractions classification AGLO.....	80
Fig.5.2. An instance of the fractions classification exercise	80
Fig.5.3. The scenario for the common divisor AGLO.....	81
Fig.5.4. An instance of the fractions classification exercise	82
Fig.5.5. An instance of the fractions amplification exercise	83
Fig.5.6. The question section for the simplification of an ordinary fraction AGLO ...	84
Fig.5.7. An instance of the fractions simplification AGLO with a wrong answer	84
Fig.5.8.The scenario for comparing ordinary fractions AGLO	85
Fig.5.9. The scenario for comparing periodical fractions AGLO	86
Fig.5.10. The scenario section for adding two fractions with common denominator	87
Fig.5.11. An instance of the fractions addition AGLO with a wrong answer	88
Fig.5.12. The scenario for the addition of two common fractions AGLO	89
Fig.5. 13. An implementation of the addition of two common fractions AGLO.....	90
Fig.5.14. The scenario section for adding two mixed periodic decimal fractions	90
Fig.5.15. An implementation of adding two mixed periodic decimal fractions AGLO	91
Fig.5.16. An implementation of the subtraction of two common fractions AGLO	92
Fig.5.17. The symbols for the multiplication of two common fractions AGLO.....	93
Fig.5.18. An implementation of the multiplying of two ordinary fractions AGLO.....	94
Fig.5.19. An implementation of the multiplying of two decimal fractions AGLO.....	95
Fig.5.20. The scenario of the fraction power AGLO	96
Fig.5.21. Two implementations of the fraction power AGLO	97
Fig.5.22. The symbols for the division of two common fractions AGLO	98
Fig.5.23. An implementation of the fraction division AGLO	98
Fig.5.24. An implementation of the fraction division AGLO	99
Fig.5.25.The scenario for finding a percentage of a whole AGLO	100
Fig.5.26. An implementation of the finding a percentage of a whole AGLO	100
Fig.5.27.The scenario for transforming a simple decimal fraction into an ordinary fraction AGLO	101
Fig.5.28. An implementation of transforming a decimal fraction into a simple fraction AGLO	102
Fig.5.29.The scenario for transforming a simple decimal fraction into an ordinary fraction AGLO	103
Fig.5.30. An implementation of transforming a simple periodic decimal fraction into an ordinary fraction AGLO	103
Fig.5.31. The scenario for transforming a mixed periodic decimal fraction into a simple fraction AGLO	104
Fig.5.32. An implementation of introduction of the wholes into a fraction AGLO ..	105
Fig.5.33.The scenario for extracting the wholes from a fraction AGLO.....	106
Fig.5.34.The theory, question and answers sections for extracting the wholes from a fraction AGLO	106
Fig.5.35. An implementation of the intersection of two intervals AGLO.....	108
Fig.5.36. An implementation of the union of two intervals AGLO	109
Fig.5.37. An implementation of the difference of two intervals AGLO	110

Fig.5.38. An implementation of equations involving absolute values AGLO	111
Fig.5.39. An implementation of Inequations involving absolute values AGLO	112
Fig.5.40. An implementation of inequation with an interval left unbounded and right-bounded as solution AGLO	113
Fig.5.41. An implementation of inequations of the form $-a * x + b \leq c$ AGLO	115
Fig.5.42. An implementation of inequations of the form $a * x + b \geq c$ AGLO	116
Fig.5.43. An implementation of double inequations AGLO	117
Fig.5.44. An implementation of abbreviated calculation formulas AGLOs	118
Fig.5.45. An implementation of decompositions into factors AGLOs	119
Fig.6.1. The question and answer of the linear search definition AGLO	122
Fig.6.2. The scenario section of the linear searching algorithm steps	122
Fig.6.3. An implementation of the linear searching algorithm steps.	123
Fig.6.4. The implementation of the operating principle of the linear searching algorithm AGLO	124
Fig.6.5. An implementation of the practical situation in which the linear search algorithm is used AGLO	124
Fig.6.6. An implementation of applying the linear search on tables of basic data types AGLO	125
Fig.6.7. An implementation of the application of linear search on different random data tables AGLO	126
Fig.6.8. The result of the searching algorithm AGLO	127
Fig.6.9. An implementation of the result of the LINEAR searching algorithm APPLIED on a linked list AGLO	127
Fig.6.10. An implementation of the recognition of an implementation of linear search AGLO	128
Fig.6.11. The question of the recognition of an implementation of linear search on a table AGLO	129
Fig.6.12. An implementation of the recognition of an implementation of linear search on a table of structures AGLO	129
Fig.6.13. An implementation of the recognition of an implementation of linear search on a linked list AGLO	130
Fig.6.14. An implementation of the searching algorithm with sentinel recognition AGLO	132
Fig.6.15. An implementation of the recognition of the position of the sentinel AGLO	133
Fig.6. 16. An implementation of the recognition of the sentinel value AGLO	133
Fig.6.17. The symbols from the scenario of the comparison between linear search with sentinel and the classical linear search AGLO	134
Fig.6.18. An implementation of the comparison between linear search with sentinel and the classical linear search AGLO	135
Fig.6.19. An implementation of the role of the binary search AGLO	136
Fig.6.20. The scenario section for situations in which binary search is applied AGLO	137
Fig.6.21. Cases in which binary search is used	137
Fig.6.22. An implementation of cases in which binary search is used AGLO	138
Fig.6.23. An implementation of cases in which binary search can't be used AGLO	139

Fig.6.24. The theory and the question sections of an implementation of binary search steps AGLO	139
Fig.6.25. The answer section of an implementation of binary search steps AGLO	140
Fig.6.26. The feedback section of an implementation of binary search steps AGLO	140
Fig.6.27. The scenario section for the binary search recognition AGLO	141
Fig.6.28. An implementation of binary search recognition AGLO	142
Fig.6.29. An implementation of search by interpolation AGLO	143
Fig.6.30. An implementation of practical situations in which interpolation search is used AGLO	144
Fig.6.31. An implementation of types of data to which interpolation sorting is applied AGLO	144
Fig.6.32. An implementation of strings on which interpolation search cannot be used AGLO	145
Fig.6.33. An implementation interpolation search steps AGLO.....	146
Fig.6.34. An implementation of recognition of the interpolation search AGLO.....	147
Fig.6.35. An implementation of insertion sorting definition AGLO	148
Fig.6.36. An implementation of practical situations to which insertion sorting applies AGLO	149
Fig.6.37. The scenario section of types of data collections on which insertion sorting applies AGLO.....	150
Fig.6.38. An implementation of types of data collections on which insertion sorting applies AGLO.....	150
Fig.6.39. An implementation of insertion sorting recognition AGLO.....	151
Fig.6.40. An implementation of insertion sorting steps AGLO.....	152
Fig.6.41. An implementation of selection sorting algorithm use situations AGLO .	153
Fig.6.42. An implementation of data types on which the selection-sorting algorithm can be applied AGLO	155
Fig.6.43. An implementation of data collections on which the selection sorting algorithm can be applied AGLO.....	156
Fig.6.44. An implementation of the Selection Sorting Algorithm Steps AGLO	156
Fig.6.45. The scenario section of the bubble-sorting algorithm steps AGLO	157
Fig.6.46. An implementation of the bubble-sorting algorithm steps AGLO	158
Fig.6.47. An implementation of the bubble-sorting algorithm interchanges AGLO	159
Fig.6.48. An implementation of the Series in Shell Sorting Algorithm AGLO	160
Fig.6.49. An Implementation of the Sequence of States for the First Series in Shell Sorting AGLO	161
Fig.6.50. An implementation of the index and value of the pivot element for quick sort AGLO	162
Fig.6.51. An implementation of identify the first element bigger than the pivot in quick sorting algorithm AGLO	163
Fig.6.52. An implementation of identify the first element smaller than the pivot in quick sorting algorithm AGLO	164
Fig.6.53. An implementation of the first exchange in Quick Searching Algorithm AGLO	164
Fig.6.54. An implementation of the quick sorting algorithm steps AGLO	165
Fig.6.55. An implementation of the adding a node in an ordered linked list AGLO	166

Fig.6.56. An implementation of the determining the position in which a node is added in an ordered linked list AGLO	167
Fig.6.57. An implementation of the successor and the predecessor of an element in the linked list AGLO.....	168
Fig.6.58. The scenario of the addition to the beginning/end of a random linked list AGLO	169
Fig.6.59. An implementation of the addition to the beginning/end of a random linked list AGLO	169
Fig.6.60. An implementation of the addition after/before an element in a random linked list AGLO	170
Fig.6.61. An implementation of the delete an element from a random linked list AGLO	171
Fig.6.62. An implementation of the deletion after/before an element in a random linked list AGLO	172
Fig.6.63. The scenario section of the adding a node to the beginning or end of a double linked list AGLO	173
Fig.6.64. An implementation of the addition to the beginning/end of a double linked list AGLO	173
Fig.6.65. An implementation of the addition after/before an element in a double linked list AGLO	174
Fig.6.66. An implementation of the delete an element from a double linked list AGLO	175
Fig.6.67. The scenario of the recognition of a tree AGLO	177
Fig.6.68. An implementation of the recognition of a tree AGLO	177
Fig.6.69. An implementation of the recognition the tree root AGLO	178
Fig.6.70. An implementation of the calculation of a tree height AGLO	179
Fig.6.71. An implementation of the distribution of nodes of a tree by levels AGLO	180
Fig.6.72. An implementation of the degree of a node of a tree AGLO	181
Fig.6.73. An implementation of the degree of a tree AGLO	183
Fig.6.74. The theory and the question of the degree of all nodes of a tree AGLO .	184
Fig.6.75. The answer and the feedback of the degree of all nodes of a tree AGLO	185
Fig.6.76. The scenario of the pears node – parent node of a tree AGLO	185
Fig.6.77. The question and the answer of the pears node – parent node of a tree AGLO	186
Fig.6.78. An implementation of determination the root nodes of each subtree of a node AGLO.....	187
Fig.6.79. The scenario of traversing the nodes of a tree in inorder AGLO	188
Fig.6.80. An implementation of traversing the nodes of a tree in inorder AGLO ...	189
Fig.6.81. An implementation of traversing the nodes of a tree in preorder AGLO .	189
Fig.6.82. An implementation of traversing the nodes of a tree in post order AGLO	190
Fig.6.83. The scenario of the recognition of a graph AGLO	191
Fig.6.84. An implementation of the degree of a graph AGLO.....	192
Fig.6.85. The theory section of the minimum path from the first node to all other nodes AGLOs.....	193
Fig.6.86. The question section of the minimum path from the first node to all other nodes AGLOs.....	194

Fig.6.87. The answers and feedbacks sections of the minimum path from the first node to all other nodes AGLOs.....	195
Fig.6. 88. The theory and question sections of the adjacency matrix AGLO.....	196
Fig.6. 89. The answers and feedbacks sections of the adjacency matrix AGLO	197
Fig.6. 90. The theory and question sections of the adjacency structure AGLO	198
Fig.6. 91. The answers and feedbacks sections of the adjacency matrix AGLO	199
Fig.6.92. The scenario for the command to create a directory AGLO	199
Fig.6.93. An instance of the command to create a directory AGLO.....	200
Fig.6.94. The scenario for the command to create a hierarchy of directory AGLO	201
Fig.6.95. The theory section for the command to create a hierarchy of directory using -p option AGLO	201
Fig.6.96. The question and answer sections for the command to create a hierarchy of directory using -p option AGLO	202
Fig.6.97. The scenario for the command to create a hierarchy of directory without -p option AGLO.....	203
Fig.6.98. The question and answer section for the command to create a hierarchy of directory without -p option AGLO	204
Fig.6.99. The feedback section for the command to create a hierarchy of directory without -p option AGLO.....	205
Fig.6.100. An instance of the command to create a file AGLO	206
Fig.6.101. An instance of the command to populate a hierarchy of directory AGLO	207
Fig.6.102. The scenario of the command to create and populate a hierarchy of directory AGLO	208
Fig.6.103. The question and answer section for the command to create a hierarchy of directory without -p option AGLO	209
Fig.6.104. The question and answer section for the command to create a hierarchy of directory without -p option AGLO	210
Fig.6.105. An instantiation of setting the file access permissions AGLO	211
Fig.6.106. An instantiation of the mount command AGLO.....	212
Fig.6.107. An instantiation of the command to start a display process in the background AGLO	213
Fig.6.108. The theory and question sections for the command to kill a process AGLO	214
Fig.6.109. The answer and feedback sections for the command to kill a process AGLO	215
Fig.6.110. An instantiation of the command to kill a process in the background AGLO	216
Fig.6.111. An instantiation of the command to move a process from background to foreground AGLO	217
Fig.6.112. An instantiation of using the options of the fdisk command AGLO	218
Fig.6.113. An instantiation of the command to view a partition AGLO	219
Fig.6.114. An instantiation of the apt command AGLO	220
Fig.6.115. An instantiation of the yum command AGLO	221
Fig.6.116. An instantiation of the command to add a user AGLO	222
Fig.6.117. An instantiation of the command to add a user to a group AGLO	223

Fig.6.118. An instantiation of the command to generate information about a user AGLO	224
Fig.6.119. An instantiation of the command to change the password of a user AGLO	225
Fig.6.120. An instantiation of the command to delete a user AGLO.....	226
Fig.6. 121. An instantiation of the ifconfig command AGLO	227
Fig.6.122. An instantiation of the command to delete an address AGLO	228
Fig.6.123. An instantiation of the command to configure an address AGLO.....	229
Fig.6.124. An instantiation of the command to activated/deactivated an interface AGLO	230
Fig.6.125. An instantiation of the route command AGLO	231
Fig.7.1. DSEL platform.....	234
Fig.7.2. An AGLO Instantiation on DSEL Platform	235
Fig.7.3. The LO repository with the students answers	236
Fig.7.4. The constructor functions for fractions objects	238
Fig.8.1. Contingency table: false positive, false negative, true positive, and true negative values of the grades.....	246
Fig.8.2. The degree of use of gadgets by students	247
Fig.8.3. The degree of interest of the students on the LOs	247
Fig.8.4. The confusion matrix	249
Fig.8.5. The questionnaire answers to the second and third questions	250
Fig.8.6. The questionnaire answers to the last two questions.....	251

1. INTRODUCTION

1.1. Thesis context

The e-learning term has become an important concept in the evolution of education. Broadly speaking, elearning (or e-learning) means the totality of educational situations in which the means of Information and Communication Technology (ICT) are used significantly [1]. The term was taken from the Anglo-Saxon literature, being extended from the primary etymological sense of learning by electronic means, and now covering the intersection of educational actions with modern computer resources. The computer and electronic/multimedia materials are used as support in teaching, learning, evaluation, or as a means of communication.

Currently, the e-learning term has practically replaced all the terms that designate a new way of integrating ICT in the training process.

Digital technology produces certain changes in the learning environment through:

- the virtual learning space, which creates the possibility of students' imagination and creativity, the information can be visualized, simulated procedural, making these materials more attractive and easier to understand;
- independent learning, students do not spend time searching for information, but using it;
- the individualization of learning, the personal pace of each one is respected;
- real-time monitoring of learner's activity.

Learning objects (LO) have emerged as a new way of thinking about learning content. Traditionally, the content comes in a matter of hours. Learning objects are much smaller learning units, lasting even just a few minutes. The concept of learning objects is broadly defined. The Institute of Electrical and Electronics Engineers (IEEE) standardization defines learning objects as: "a learning object is defined as any entity, digital or non-digital, that may be used for learning, education or training" [2].

There is a variety of learning object models and new proposals are continuously appearing. The most widely used e-learning standards are

- Sharable Content Object Reference Model or SCORM [3];
- The Institute of Electrical and Electronics Engineers Learning Object Metadata or IEEE LOM [4];
- Information Management System or IMS [5];
- Learning Resource Metadata Initiative or LRMI [6];
- Computer Information System Company or CISCO [7].

A learning management system (LMS) is an adapted environment for disseminating learning objects. It is a software application that provides, among other things, educational courses and learning programs. The teacher can use this software to create structured courses and manage them to meet various requirements. LMS is considered to be the foundation for building today's e-learning practice [8]. Some open-source LMS are Moodle [9], Ilias [10], Canvas [11].

LOs are difficult to reuse and difficult to adapt to each discipline, as it requires access to source code, programming knowledge for content modification, testing, and

deployment [12]. To improve the use of learning objects, the generative model was created. "Generative Learning Objects (GLOs) are generic and reusable LOs from which the specific LO content can be generated on-demand" [13].

In the field of generative objects there are two clear directions:

- GLO based on templates, developed by Tom Boyle in [14], [15], [16].
- GLO based on meta-programs developed by Damaševičius and Štuikys in [17], [18], [13].

Auto-generative learning objects (AGLOs) are "reusable pedagogical patterns to be instantiated with generated content based on random numbers to fulfill the learning objectives" [12]. In this thesis, we propose an AGLO based approach to facilitate learning and automatic assessment for science, technology, engineering, and mathematics (STEM) disciplines.

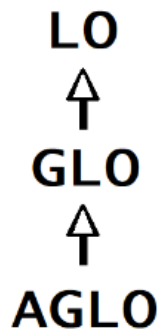


Fig.1.1. The LOs conceptual hierarchy

In Figure 1.1 we present the conceptual hierarchy for several types of learning objects: simple, general learning objects, generative learning objects, auto-generative learning objects. LOs are the digital resources with which the student interacts. GLOs are objects from which the specific LO content is generated when requested. AGLOs are templates from which concrete LOs are obtained at each instance.

For both teachers and students, it is important to find the right resources and combine them with other learning materials in order to have the best learning methods. To meet this opportunity, we have created AGLOs that can be integrated into learning management systems such as Moodle. This offers both accessibility and quality. Our approach is schematically represented in figure 1.2.

Initially, the tutor develops an AGLO model that aims at a specific learning objective. The educator develops the meta-model using accessible media: Eclipse and Notepad ++. When templates are created, they are stored in a storage unit, a MySQL database. Templates are saved in XML files. From the database, the student through the frontend web application can select AGLOs.

Students access the web application using a web browser on a workstation, tablet, or smartphone. In the assessment process, the student will access several AGLOs. At this step, the accessed AGLOs are instantiated with random numbers, the formulas are evaluated to meet the projected learning or testing scenario. Methods from domain-specific JavaScript libraries are also called. The evaluation of the correctness of the answers is done automatically. The answers given by students are stored in a repository of learning objects (LOR).

An AGLO is a pedagogical model that instantly provides different exercises so that the student can practice the notion. To exemplify, we will further present an example, namely, the addition of a node in a simple linked list. The pedagogical objective pursued is for the student to know how to add a node in a simple linked list. To create this scenario, a simple linked list and a node are randomly generated. The list has both a graphical representation, for the student to visualize the notion, and a representation in the form of a string in order to be able to represent the answer.

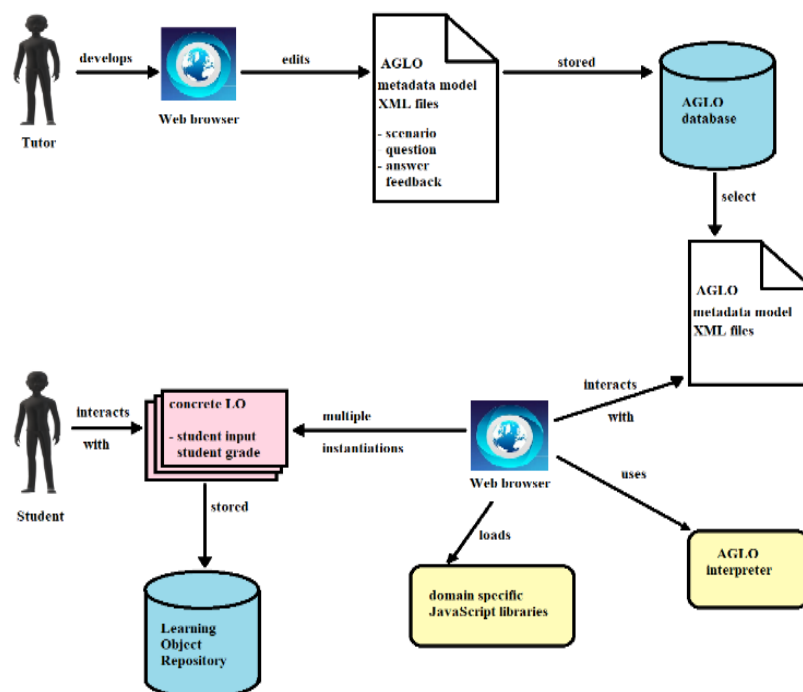


Fig.1.2. The AGLO Approach

In the concrete LO with which the student interacts are the following sections:

- the theory section, a section in which the notion is presented to the student and an example, this is static, the same for each instance, and it also has the role of showing the student in what form the answer should be written;
- the question section, in which the work task is stated, at each instantiation the workload depends on the randomly generated list and node. In the case of the example presented in Figure 1.3, an ordered simple linked list consisting of three nodes is generated, and a node will have to be inserted in the simple linked list. Depending on the value of the information in the generated node, it must be added at the beginning, at the end, or inside the linked list, consequently in the above case the addition is made at the beginning;
- the answer section, where the student writes the answer in the form of a string, respecting the model from the example presented to him. In Figure 1.3 we chose to write the correct answer;
- the feedback section is generated when the evaluate button is activated, it contains both explanations and the correct expected answer.

The field of e-learning has a significant potential for research as it is a rather new concept in Romania's current socio-economic context. Young people in our society are almost technology-dependent, using electronic devices for learning, socializing, and leisure. Learning is a process and is part of our everyday life. New modern educational technologies support individual learning that gives students the chance to learn on their own, but not exclusively. Students can learn from anywhere through online education and mobile education. This has made learning more convenient and fun.

Theory

Adding to a simple ordered linked list is done in such a way that the list remains ordered.
 For example if you want to add the node {0x01,48,0x00}
 in the ordered linked list {0xa,46,0xe}->{0xe,58,0x1a}->{0x1a,61,0x22}->{0x22,70,0x26}->{0x26,82,0x2e}
 The new obtained list is:
 {0xa,46,0xe01}->{0x01,48,0xe}->{0xe,58,0x1a}->{0x1a,61,0x22}->{0x22,70,0x26}->{0x26,82,0x2e}

Question

The following list is given:

0xa	28	0x16	—	0x16	40	0x22	—	0x22	54	0x00
-----	----	------	---	------	----	------	---	------	----	------

The node to add is {0x01,24,0xa}.
 What is the new list thus obtained?

Answers

{0x01,24,0xa}->{0xa,28,0x16}->{0x16,40,0x22}->{0x22,54,0x00}

You got 10 out of 10! Evaluate

Feedbacks

Adding to a simple ordered linked list is done in such a way that the list remains ordered.
 In this case the list thus obtained is: {0x01,24,0xa}->{0xa,28,0x16}->{0x16,40,0x22}->{0x22,54,0x00}.

{0x01,24,0xa}->{0xa,28,0x16}->{0x16,40,0x22}->{0x22,54,0x00} {0x01,24,0xa}->{0xa,28,0x16}->{0x16,40,0x22}->{0x22,54,0x00} OK

Fig.1.3. An instantiation of the addition of a node in an ordered linked list

A generative model is an effective way to learn using independent learning. Generating generative models on STEM disciplines would be an effective support for students. Interactive queries play an important role in e-learning and offer a wide range of benefits to both the student and the tutor. The model proposed by us fulfills these aspects.

1.2. Thesis objective

The research activity that is the subject of this doctoral thesis focused on the following main objective: the development and implementation of an AGLO-based approach to facilitate learning and automatic assessment for STEM disciplines. The

purpose of such an objective is that we create AGLOs that can be used in the instructive-educational process.

The study, concept, development, implementation and testing of the approach are just some of the steps taken to achieve this main goal. In view of the main objective mentioned above, a number of specific objectives have been formulated:

Objective O1 is to make a bibliographic study of LOs and a comparative analysis of the studied models;

Objective O2 is to propose a step-by-step methodology based on abstractions to create reusable AGLO templates;

Objective O3 is to abstract multiple learning concepts from STEM disciplines (Arithmetics, Data Structures, Algorithm Analysis, Operating Systems) into instantiable AGLOs (at least 150 objects);

Objective O4 is to develop domain-specific libraries modeling concepts (at least 15 JavaScript classes) to assist the instantiation of AGLOs;

Objective O5 is to validate our AGLO approach in practice, on groups of students showing their effectiveness;

Subobjective O51 is to show that AGLOs are effective in the learning process compared to classical approaches;

Subobjective O52 is to show that the AGLO's automatic assessment mechanisms are very close to the evaluation given by tutors.

1.3. The structure and content of the thesis

The content of this doctoral thesis is structured in nine chapters. For an overview, the notes covered in each chapter will be presented below.

Chapter 2 presents the state of the art in the field of e-learning and LOs. The chapter begins with a review of the main LO models. The chapter continues with the exposition of LMS platforms. Next, the chapter presents the new generation of LOs, namely GLOs. Other methods for improving LO through sequencing, object-oriented programming, and integration into cloud frameworks are presented. Finally, a comparison of the presented models is made.

Chapter 4 presents step by step the abstraction process. The abstraction process is then presented practically on an example from the disciplines Mathematics, Data Structures and Algorithms, Algorithm Analysis and Design, and Operating Systems. The chapter further presents the structure of the model. Finally, the model semantics are presented.

Chapter 5 presents the AGLO model applied in the field of middle school arithmetic. The chapter includes the presentation of the AGLOs aimed at working with fractions, operations with intervals, solving equations and inequations involving absolute values, solving different types of inequalities, as well as abbreviated calculation formulas.

Chapter 6 presents the AGLO model applied in the field of IT disciplines. The chapter begins with an exposition of AGLOs targeting four types of searches, namely linear search, linear search with sentinel, binary search, and interpolation search. The chapter continues with the presentation of the AGLOs that aim at different types of searches. Working with linked lists and double linked lists is the topic discussed later in this chapter. Notions related to trees and graphs are further addressed. Finally, the AGLOs have presented that aim at notions related to different basic commands used in Linux.

Chapter 7 presents the prototype implementation. The chapter presents how we achieved to connect the students with our AGLOs using the DSEL platform. The domain-specific Java Script libraries we made are further set out.

Chapter 8 presents the validation of the AGLO model. The chapter presents two case studies, one made on a group of 12 fifth grade pupils, and one made on a group of 50 eighth grade pupils.

Finally, Chapter 9 presents the final conclusions, the original contributions from the doctoral thesis, the dissemination of the results, as well as the direct research future.

2. STATE OF THE ART IN LEARNING OBJECTS

This chapter provides a theoretical basis for the design and development of LOs. The chapter presents definitions of specific terms, describes the preconditions for implementing the design and development of LOs, explains the role and purposes of LOs, as well as the need to use them. It justifies the principles considered in this research as a basis for the design of AGLOs and their use in the educational process.

2.1. Learning Objects Definitions

There are several definitions of the concept of "learning object" since different directions are followed: pedagogical, technical, and economic. Consequently, some formal definitions are accepted. There are some generally assumed requirements for LOs:

- objects should be independent of the sharing system;
- objects should be reusable, not dependent on the educational context;
- objects should have a certain level of aggregation to be able to combine them;
- objects should be characterized by suitable metadata.

In the initial stage, a learning object was defined as "a collection of content items, practice items, and assessment items that are combined based on a single learning objective" [7].

The term is attributed to Wayne Hodgins and was first used in a working group on Applied Science and Technological Progress in 1994. This new conceptual model for creating and distributing content was seen as an important element in Hodgins' vision of the future of learning. He believes that learning objects are designed to change learning, introducing unprecedented efficiency in the design, development, and dissemination of content. Thus, their most important quality was seen, at that time, as the improvement of learning efficiency and human performance [19].

In [20] a LO is defined as: "A digital self-contained and reusable entity, with a clear educational purpose, with at least three internal and editable components: content, learning activities and elements of context. The learning objects must have an external structure of information to facilitate their identification, storage, and retrieval: the metadata." Chiappe and his collaborators believed that the incorporation of information and communication technologies is the feasible path to be considered in order to improve the quality of education. The enormous potential that learning objects have in the educational scene will play a major role in strengthening the quality of education. In their view, learning objects are different from any other educational material produced by a teacher in an isolated way and should be open-source. Compared to an entire course, LOs are considerably lower and follow a modular behavior that allows easy application as study material for self-employment. They are also applicable not only as study material but also as a teaching strategy revealing pleasant surprises in the teaching-learning processes [20].

Centre of Excellence in Teaching & Learning in Reusable Learning Objects (RLO-CETL) from England defines a reusable LO as "web-based interactive chunks of

e-learning designed to explain a stand-alone learning objective" [21]. CETL focused on providing pedagogical and structural design principles for creating learning objects.

In general, LOs are "digital resources that can be used (and reused) to support the learning process" [13].

Others as seen in [22] define it as "a digitized entity which can be used, re-used or referenced during technology supported learning".

In this context, there is an international interest in developing good models of reusable learning objects. For this purpose, there are funding programs for projects that address this area, such for example The Joint Information Systems Committee Design for Learning (D4L) program, ALTC Competitive Grant "Implementing Effective Learning Design". There are a series of views on the learning model eg. [2], [14], [23], [24].

All projects vary according to the studied unit, chapter and object, and what the designer was following. These variations may lead to the creation of duplicates, which is why a conceptual model is needed to link these learning patterns [25].

LOs were developed as part of a learning environment. Individually, each LO was created for a specific educational purpose. Several aggregate objects could become a lesson and thus fulfill a pedagogical objective.

The idea behind the use of LO is based on the following: discoverability, reusability, and interoperability. Discoverability is accomplished by the fact that LOs are described using metadata, for example, IEEE Standard. Reusability is supported by several specifications, such as the IMS Content Package. Interoperability is supported by a model like the one developed by ADL Organization, namely SCORM.

For facts, the idea is that learning objects are seen as atoms, small fragments that can be organized together. Standardization bodies have perfected this concept, have specified information on how to sequencing, and have provided information on their organization in courses, as well as assembling them for delivery.

From an economic point of view, LOs have been designed to reduce learning costs, standardize content, and support profitable content reuse through learning management systems [4].

LOs are stored in repositories (which are data structures). There are two types of deposits: containing both LO and their metadata, and those containing only metadata. These repositories can be:

- centralized - stores LOs metadata on a single server or website and links to LO,
- distributed - metadata are stored on multiple linked servers or websites.

Besides storing and recovering data, a repository also aims to share and reuse data.

A learning object is a module compatible with e-learning rules, which intends to support the instructive-educational process, very specific (aims at a learning objective), and autonomous. Learning objects are stored in the content management system, for just-in-time learning in different contexts of a learning process.

2.1.1. IEEE Learning Object Metadata Standard

The Institute of Electrical and Electronics Engineers Standards Association (IEEE) has achieved an internationally recognized open standard that has been several versions and is called the IEEE Standard for Learning Object Metadata (IEEE LOM). The IEEE Learning Technology Standards Committee (LTSC) has described the metadata learning object model [4].

The purpose of this model is to facilitate the use and distribution of learning objects. This standard can be easily used for both sharing and exchanging of learning objects. The model also aims to reduce the cost of delivering high-quality resource services through sharing between systems for discovering learning resources.



Fig.2.1. The hierarchy of elements in the LOM data model

The model can provide information in a standard format if tutors choose to label resources according to some specifications. Resource descriptions are tailored to needs and include both the choice of vocabulary and the reduction of the number of items that are described or the addition of new ones.

The IEEE LOM data model identifies clearly and definitely which properties of a learning object must be described and what vocabulary should be used for descriptions. Specifications also define how the pattern can be modified by additions or constraints. There are also parts of the standard developed to define LOM data model links, in other words, LOM records are represented in XML and RDF.

Metadata is the information about an object. Due to a large number of objects, the existence of metadata is vital. The standard tackles this issue. IEEE LOM is based on defining a structure for describing metadata for a LO.

The model includes a hierarchy of elements, on the first level there are 9 categories, which in turn are made up of other sub-categories. The hierarchy is shown in Figure 2.1. As can be seen, a metadata instance describes the relevant characteristics of an LO, and these features are organized into the following categories:

- The General category includes general LO information,
- The Lifecycle category describes the history and the current state of LO,
- The Meta-metadata category contains information about the metadata instance itself,
- The Technical category describes the technical characteristics of LO,
- The Educational category describes the educational and pedagogical characteristics of LO,
- The Rights category describes LO's property rights and terms of use,
- The Relations category comprises defining relationships between different LOs
- The Annotation category offers annotations about LOs, but also information about whom and when they created the comment.
- The Classification category reports this LO to a classification system.

This classification represents the LOM outline. For efficient semantic interoperability, "extended data elements should not replace data elements in the LOM structure" [24].

The model specifies that a classification item can be repeated, thus allowing it to be used for various purposes. Also, specify value space and data type for each of the data elements. The value space defines what data can be included for that element: a string, elements from a declared list, or elements from a specific format. Data types may contain a string of characters or two parts:

- LangString items contain Language and String parts, which render the information in several languages;
- Vocabulary items are chosen from a list of terms, each element being consisting of the name of a list of terms used and the chosen term.

Date Time and Duration items contain two parts, one having the date given in a machine-readable format, and one allowing a description of the date or duration

The structure of the metadata description is intended for portability and may be referenced by other standards. It is a base that can be expanded according to the tutor's needs, so it can be easily adapted to the student's level. So this model can also be used as a basic scheme in the development of automated LOs.

Each element can be simple, so it only contains data, or it can be an aggregate element and contain other elements. The semantics of any element depends on the context, the parent element in the hierarchy, and other elements that are at the same level as it.

Any instance of metadata for a learning object has important features. This model has four levels of granularity. The smallest level is level 1 and represents raw media data or fragments. Aggregation level 2 is represented by collections of level 1 objects and is the equivalent of a teaching lesson. Level 3 is the equivalent of a course and is made of level 2 objects. Level 4 is accomplished through a collection of level 3 elements recursively through level 4 elements.

The base of this model does not define how a learning system presents or uses a metadata instance for a LO. When implementing LOM as a data or service provider, creating an application profile allows tutors to specify the elements and vocabulary to use. Elements can be replaced with others from other metadata, and vocabularies can be enriched with personalized elements specific to the goals being pursued.

The most important things to be followed when using this model are:

- understanding the needs of those who will use learning objects;
- creating an effective strategy to create high-quality objects;
- stacking in a form that allows for efficient portability;
- communicate with other systems by exchanging basic information.

This model of learning objects supports both active learnings through content that requires the student to engage in productive actions or decisions, as well as passive learning through materials that have the role of exposing a particular learning content.

Objects made with IEEE LOM can be transferred between systems using a variety of protocols, the most widely used being OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting). As a resource on the Internet CanCore is currently the most comprehensive element-to-element guide to LOM existence.

The Standards Development Lifecycle is a process in six stages:

- Initiating the Project, projects are started when there is a need for an idea or concept to be standardized;
- Mobilizing the Working Group, the group must operate in compliance with the IEEE-SA Standards Board Bylaws;
- Drafting the Standard, each project version must respect copyright;
- Balloting the Standard, this begins when the Sponsor is satisfied with the standard version;
- Gaining Final Approval, based on the recommendation the Standards Review Committee the IEEE-SA Standards Board approves or disapproves standards;
- Maintaining the Standard, a standard is valid for ten years from the date of its approval.

2.1.2. Sharable Content Object Reference Model

The Sharable Content Object Reference Model (SCORM) consists of assets, Shareable Content Objects (SCOs), and aggregates of content. An asset can be a text, a picture, an educational movie, an audio recording, a webpage that can be represented on a client page. SCO is a collection of one or more assets and can be reused in different learning contexts. Content aggregation is a well-structured structure that can be used to organize learning resources in a course, chapter, or module [24].

When an SCO contains multiple HTML pages, it is responsible for providing the navigation interface for those pages.

SCORM Content Packaging contains three types of components: two are considered resources: assets and content objects that can be shared, and the third is content aggregation. These components can be grouped into packages that can contain video streams, text documents, and others [24].

Metadata is added to these packages to allow them to be searched and reused. Metadata can be defined at any level of aggregation in the content. SCORM uses IEEE LOM as a metadata profile, which includes nine metadata categories to describe an object: general, life cycle, meta-meta data, technical, educational, rights, relationship, annotation, and classification [26].

Shared content objects form "launchable learning resources" [26]. These objects are compatible with SCORM compliant learning management systems. The courses consist of both packages and content objects.

Learning objects are considered to be the smallest semantic entities. Learning objects can be aggregated to form a course or other more complex learning objects.

Courses can be seen as complex learning objects from an object-oriented vision. The design of the courses is done by grouping the learning objects in packages that respect the Instructional Management System standard [27].



Fig.2.2. An example of a SCORM course

A SCORM course is like a presentation but with an additional level of interactivity. An example of such a course is shown in Figure 2.2.

Regarding the realization of a course, it is necessary to take into account the fact that the aggregations form the logical hierarchy of the course.

The student can have a personalized course. Depending on his level of knowledge he may be allowed to omit some exercises from that course. How a student can navigate between the components of a course is determined by a set of rules and attributes that are found in a manifest XML file.

It is necessary to use typed resources as assets instead of SCOs, as they do not communicate with LMS.

A JavaScript controller is added at runtime to manage SCO browsing. This controller has the role of

- mark the current location of the learner,
- report progress through content,
- to record the total time that the student spent in training,
- report the score based on test results.

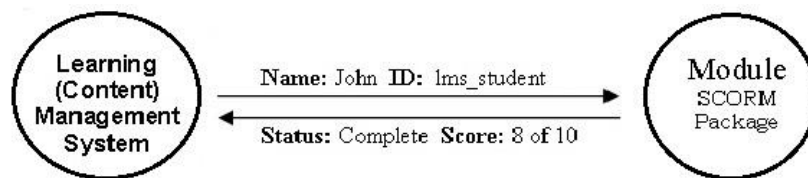


Fig.2.3. How LMS returns student status

The result that a student can get in the course can be: incomplete, completed, passed, or failed, see Figure 2.3.

This model offers a high level of flexibility as it does not require the module to be organized in a certain way. This level of flexibility is also because content protection is imposed, that content may have copyright restrictions. SCORM facilitates the provision of a very general model on which to build a specific learning object. This feature is included in technical interoperability. However, in this model the problem of the size of a basic object is not solved, the granularity remaining quite vague in this context.

2.1.3. Cisco Learning Object Model

The original reusable learning object (RLO) strategy was defined in 1998 by a cross-functional team at Cisco Systems. Like ADL and IMS, Cisco is actively involved in developing models to meet current challenges that meet the demands and feedback of partners and collaborators [7]. From the first version to the present, Cisco has so far improved its definition of the learning object, as well as its implementation strategy.

Cisco has published a strategy for developing and implementing RLOs. For the Cisco team, an effective LO aims to achieve a single learning objective, so this RLOs focus on a single learning objective or performance. An RLO consists of a collection of practical activities, static or interactive content, and also includes evaluations to measure the achievement of the learning objective which it targets. Evaluations can be found within the LO or placed separately as an assessment group. The components are made up of raw media assets: text, animation, audio, video, Flash, Java code, applets, and any other structure needed for the given case. LOs are identified with metadata so that they can be easily referenced and searched by both tutors and students.

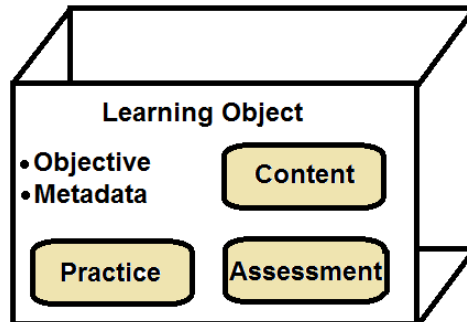


Fig.2.4. The CISCO Learning Object structure

The learning object structure is shown in Figure 2.4. Each object includes an overview of the content, lesson summary, practice, and assessment. To be conveniently grouped to achieve an effective hierarchy such as a lesson, module, course, or curriculum, LO has a granular structure. The model is designed with a simple two-level hierarchy consisting of an RLO and reusable informational objects (RIOs).

From a structural point of view, an RLO is a collection of 7 ± 2 RIOs. To achieve a complete learning experience from a collection of RIOs, a summary and evaluation are added.

RIOs are information built around a learning objective. Each RIO is made up of three items: content, practice, and assessment. A practical element is an activity that allows the student to apply his / her knowledge and skills, such as a case study or a practical activity. An element of assessment is a measurable problem or activity to determine if the learner has achieved the targeted learning objective.

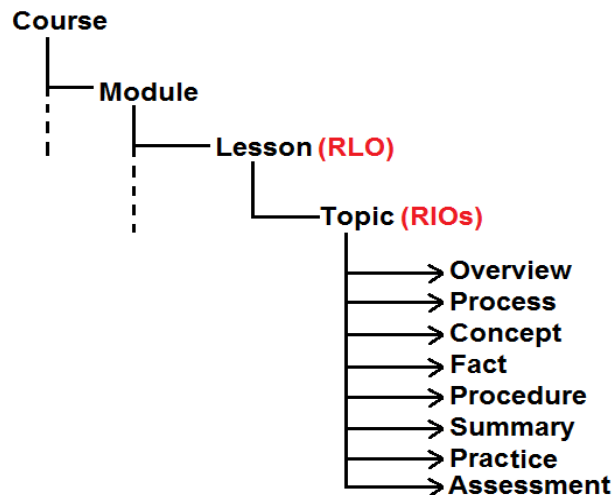


Fig.2.5. Course Hierarchy in CISCO Learning Object Model

The full hierarchy of a course is depicted in Figure 2.5. Lesson topics are based on the five types of information defined by Ruth Clark in *Developing Technical Training* and adapted and used by Cisco Systems, namely concept, action, process, principle, and procedure. Using this model provides a template for creating the content, practice, and evaluation that forms the learning object. As you can see a lesson includes a general overview, summary, practice, and evaluation.

Among the benefits that tutors benefit from by adopting this strategy are: they have significant support in designing multiple learning approaches, they can combine old and new objects to build new solutions to meet student needs, they save time and resources through objects "published" for delivery. Students also gain from using this mode, among the benefits they are: they can self-assess their abilities and knowledge, have multiple types of support and presentation styles, and multiple learning approaches (passive learning, learning by discovery, solving problems).

As you can see, there are many benefits to adopting an RLO strategy, but there are also challenges. These can be tools and systems, or writing styles and guidelines. Some challenges may be quite significant, even limiting the scope of the objects. In recent years, following feedback, Cisco has focused more on "repurposing" content instead of "reuse" [7]. The authors find learning to be a useful base for the content they can convert to fit their needs.

The learning object design and development model is a set of repetitive processes that are used to design and develop any solution that responds to the needs of students. This model allows you to define the scope of the project, meet milestones, and evaluate the success of the solution. The design and development model begins with training needs analysis, passes through the design and development stages, and ends with evaluation and delivery. The result should be a learner experience that responds to the student's needs.

"A learning system supports the transfer of knowledge and skills far beyond the traditional events scheduled in time (such as a class)" [7]. In the learning object development process, the primary change is the ability to reuse and repurpose existing LOs.

The learning object development process has some stages:

- Granular Analysis - this stage of the learning object development process examines all the factors that influence the needs of the target audience, which causes the difference in performance, identifies the desired results, and uses this information to select the best intervention;
- Design and Mine - at this stage, a training solution is structured, learning objectives are captured, content types are identified, and tutors agree that the solution responds to identified needs in the analysis phase, learning objects are now identified that either fit the needs without changes or can be repurposed;
- Reuse and Develop - this stage focuses on developing all the resources, content, and interactions for each LO;
- Delivery and Reference - at this stage, learning objects are prepared in a delivery environment as a learning experience;
- Maintain for Life - any learning object made available to students is updated and maintained for the entire "lifecycle";
- Evaluate.

In the second stage, the idea of "mining" for learning objects is important, so there are looking for solutions that could exist and are already used by the student. An important property of a LO database is its ability to determine what has already been created, to add new items to existing objects, and produce statistics on use or evaluation. Adaptation and design depend on the existing creation and delivery systems and how familiar the tutor is to access data about the use of a particular LO.

It is preferable to create an integrated database that allows the design of the LO to be automated. Such an instrument allows the tutor to move from the design phase to the development phase with or without a formal print delivery. Given the granular nature of LOs, it is possible to design and approve individual learning objects independently of the larger structure of the course.

The RLO strategy can be customized; there are different types of learning objectives, depending on the level of the learning experience and its associated level in a hierarchy of the RLO directive. For example, as you advance in the hierarchy from lesson to subject, discover that learning objectives become more specific until you reach what is known as a "terminal goal." This goal is the measurable result that results from the fact that the student completes with success LOs.

In terms of classifying learning objects, those who use CISCO use learning objectives to classify each subject. Five basic types are defined: concept, fact, procedure, process, and principle. A concept is a group of symbols, events, or ideas that are defined by a word or a term, which have common features and differ in irrelevant characteristics. A fact is unique, specific information in the form of a statement, images, or specific data. A procedure is a sequence of instructions that repeats the same way each time and is to be followed to perform a task or make a decision. A process is a flow of events, not necessarily done individually, that describes how something works. Principles are the directions for tasks adapted to different situations that provide students with guidelines for action.

During the development of learning objects according to this model, it is crucial to include metadata about each developed LO. Some metadata should be captured and automatically maintained by the system (author, data, support type, hierarchy, size, etc.). A LO is incomplete without metadata because most of the benefits of the RLO strategy would be lost without using them.

2.1.4. Dublin Core Standard

Dublin Basic Metadata Element Set (Simple Dublin Core, standardized as ANSI / NISO Z39.85-2001) provides a simpler set of elements that overlap with IEEE LOM and is useful for sharing metadata across a wide range of disparate services [28]. Dublin Core was first published in a 1995 workshop report and consisted of thirteen elements. In 1998, this was formalized in the RFC 5791 standard of the Internet Engineering Task Force and discussions began on its transformation into a standard of the (US) National Organization for Information Standards (NISO).

The Dublin Core Metadata Initiative works on a set of terms that allow the Core Dublin Set to be used with Core Qualified Dublin Core. Dublin's Working Group on Education aims to provide Dublin Core for the specific needs of the educational community. Details about Dublin Core can be found on organization web site. The goal of the standard is to discover resources for any networked resources.

The Dublin Core Metadata Element Set contains data divided into fifteen categories. These were organized into three main categories: Dublin Core Content, Intellectual Property, Instantiation.

Dublin Core Content contains the following types of resources:

- Coverage, renders the spatial or temporal subject of the resource, the spatial applicability of the resource or the jurisdiction under which the resource is relevant;
- Description, specifies a resource account;
- Type, describes the nature or type of the resource;
- Relation, gives a related resource;
- Source, represents a related resource from which the described resource is derived;
- Subject, specifies the subject of the resource;
- Title, represents the name given to the resource.

Intellectual Property includes

- Contributor, represents an entity responsible for contributions to the resource;
- Creator, represents an entity primarily responsible for making the resource;
- Publisher, represents an entity responsible for making the resource available;
- Rights, reproduce information about the rights held in and over the resource.

Instantiation comprises the following types of resources

- Date, provides a point or time period associated with an event in the resource's life cycle;
- Format, provides information about file format, media, or resource size;
- Identifier, renders an unequivocal reference to the resource in a given context;
- Language, specify the language of the resource.

```
<meta name="DC.Format" content="video/mpeg" />
<meta name="DC.Language" content="en" />
<meta name="DC.Publisher" content="prof. Costea" />
<meta name="DC.Title" content="Insert sort" />
```

Fig.2. 6. Dublin Core metadata encoding examples

The syntax choices for metadata depend on the context. An example of encoding is shown in Figure 2.6. Concepts and semantics are syntax-independent and apply in different contexts, as long as the metadata is in a needle form can be interpreted by both devices and people.

Dublin Core metadata is not only used for simple description of resources, it can also be used to combine metadata vocabularies of different metadata standards, or to provide interoperability for metadata vocabularies in the connected data cloud.

The resources described using the Dublin Core may be physical resources such as books or CDs, as well as video, images, web pages, and objects like artworks.

2.1.5. The Learnativity Content Model

The Learnativity content model is flexible and constitutes a basis for expanding the learning content architecture [29]. This model has a fixed number of

granularity levels, and it also discusses the implicit restriction of combining objects of the same granularity.

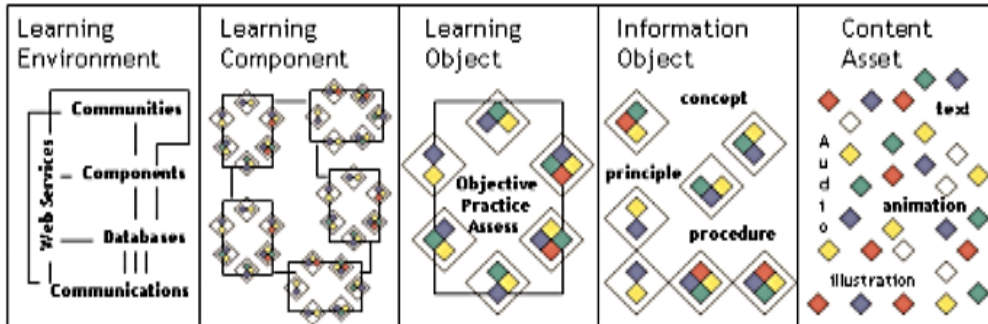


Fig.2.7. The five-level Learnativity content hierarchy

The template defines a five-level content hierarchy, see Figure 2.7:

1. Raw & Data elements are the lowest level and refer to content elements that can be a single sentence, a paragraph, an image, an animation.
2. An information object combines raw data and media elements and focuses on a single piece of information. Such content could explain a concept, illustrate a principle, or describe a process. These information items can be considered exercises.
3. Based on a single objective, the information objects are assembled in the third level of the application objects. This level is considered to contain what may qualify as a learning object. Accordingly learning objects are actually considered to be a collection of information objects, which refer to a single learning objective [29].
4. The next level refers to aggregate assemblies that deal with larger objectives. This level corresponds to the lessons or chapters.
5. Lessons or chapters can be assembled into larger collections, such as courses and curricula. These collections refer to the last level of aggregation of the model.

One can notice that the first two levels are not learning objects of their own, they cannot independently support learning. It would be more effective to use a scenario in which learning objects of different granularity can be combined to allow the student to adapt dynamically [30].

2.1.6. NETg Learning Object Model

The National Education Training Group (NETg) is a Thomson learning company, a member of the IMS Global Learning Consortium [31]. NETg was one of the first to use the concept of LO for IT courses. In 2007, Thomson sold NETg to SkillsSoft. Currently this type of learning object is used in US universities.

This company has created its own group of learning management system (LMS) developers, whose systems are designed to work with the NLO (NETg learning object) architecture.

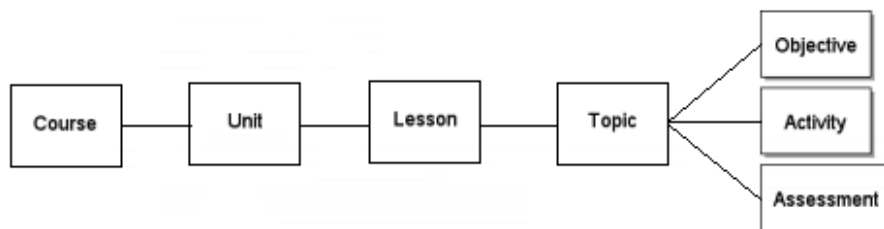


Fig.2.8. The architecture of the NETg Model

This architecture is made on a hierarchy of 4 levels - course, unit, lesson and subject, see Figure 2.8. From the NLO perspective, a course is structured as a three-dimensional matrix. Each cell of the matrix can be considered to be a subject, a column of subjects forms a lesson, and a line of lessons forms a unit.

A course contains independent units, a unit contains independent lessons, and a lesson in turn contains independent subjects. A subject is an independent learning object that contains a single learning objective and has an appropriate activity and assessment [12].

A subject is known as a NLO, which is defined as the smallest independent learning experience. Such a learning object is a single measurable or verifiable step that aims to achieve a learning objective.

Using a tool like NLO+, NETg content can be mixed and adapted from different courses to create a new course. When the student needs information, he can browse the digital library, enter an application and obtain relevant objects. If the student needs a complete course on a subject, the system will build a course based on the proposed objectives.

2.1.7. Abstract Learning Object Model

Verbet and Duval present in [32] a model that allows the generalization of other standards such as Learnativity, SCORM, or NETg. Abstract Learning Object Model (ALOCoM) is built on three levels:

- Content fragments - are individual text, audio, or video files, and these are instantiations;
- Content objects - are sets of content fragments and they are the abstract type;
- Learning objects - independent training experience.

The model is made on the levels of aggregation of objects. This defines a topology between content objects that can communicate with the outside.

ALOCoM is useful because it has a high degree of reusability. Reusability is because it does not distinguish between types of obituaries in terms of size or pedagogical significance. It also allows applications to a variety of learning environments. There is also an ontology that supports this model of ALOCOM Ontology.

ALOCoM was developed as a mapping of the NETg [31], SCORM [3], Cisco [7] and Learnativity [29] models. The authors developed a model that defines three levels of aggregation to address interoperability issues.

The paper [32] presents the new version of the model. In this model, the content of the abstract learning object was performed according to the method introduced in [33]. The method has three main steps:

- building a global ontology that covers existing content models
- building local ontologies for each content model;
- defining mappings between ontologies.

A set of content analysis categories and types of questions were defined based on seven types of information:

1. A "concept" describes a generalized abstract or generic idea of certain situations. A concept is used to teach a group of objects, symbols, ideas, or events that are designated by a single word or term, have a common feature, and may vary depending on irrelevant features.

2. A "fact" provides information based on real events; describes an event or something that holds it without being a general rule [34].

3. A "classification" is a sorting of articles into categories. A typical example is "the presentation of technologies in medical imaging" [35].

4. A "structure" is a physical object or something that can be divided into parts and has boundaries. A typical example is the "anatomy of the human brain" [35].

5. A "principle" is a basic generalization accepted as true and which can be used as a basis for reasoning or behavior [34].

6. A "procedure" consists of a specific sequence of formal steps or instructions to achieve a goal. Typical examples are the "Euclidean algorithm" or "instructions for operating a machine" [34].

7. A "process" describes a sequence of events. A process provides information about a flow of events that describes how something works and that may involve multiple actors. An example is "how a computer system responds to commands" [34].

Guidelines have been developed to identify which blocks of key information are needed to fully understand a topic.

The elements of the learning object are subdivided into:

1. Fragments of content, defined as individual components of content, such as text, images, audio and video fragments.

2. Content objects, defined as learning components that aggregate fragments of content. Content objects focus on a single piece of information and can be used to explain a concept, illustrate a principle, or describe a process.

The ALOCoM model defines content fragments at the lowest level of granularity. Content snippets are uncombined content components that represent digital representations of media such as text, graphics, animations, video, or audio.

The learning objects are divided into:

1. Learning objectives with a single objective, defined as aggregations of the components of the learning object that refer to a single learning objective. Examples are concepts, facts, principles, processes and procedures.

2. Multi-objective learning objects are aggregated of single-objective learning objects and refer to larger learning objectives. Examples are chapters and lessons.

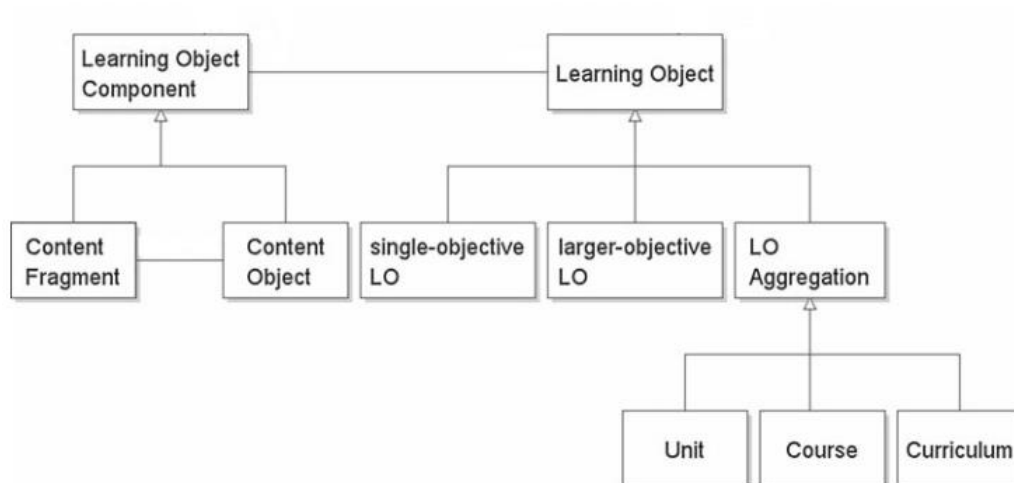


Fig.2.9. The ALOCOM model architecture

The ALOCOM model and the ontological cartographies presented in [32] are an attempt to align the content models of existing learning objects and aim to facilitate their interoperability.

The ontology connects the content model specifications that are currently available. Such ontology is never completely stable and should evolve over time. In addition, because it is an attempt to integrate different points of view, the mappings are subject to discussion.

2.1.8. H5P learning objects

H5P interactive content is an open source tool based on JavaScript, focused on creating interactive HTML5 content [36]. It is used to facilitate the creation of various interactive activities that can be integrated both in your own site and on other LMS platforms.

Interactive content can be created both directly on h5p.org and on your own device after installation. More than 40 different types of multimedia activities are available, such as word completion, timeline creation, virtual tours, memory games, and more. The tendency is to increase these variants with the opening of the source code.

Depending on the type of task, the difficulty of creating it differs. However, the process of creating interactive learning tasks does not require advanced programming knowledge. The uniform structure of tasks can be expressed using two basic components content and settings.

Content, the first component of the object, refers to texts, images, videos or audio recordings. These are inserted in predefined fields from which the workload is subsequently generated. Each task type is characterized by a specific dominant multimedia element, although several multimedia elements can be integrated into a task. The general feedback module is also included in the content component. In this module tutors can set their own assessment scale and / or verbal feedback.

The second component, Behavioral Settings, contains customizable behavioral attributes of tasks, such as whether students can perform the task repeatedly, whether incorrect answers are penalized, and so on. The individual settings vary depending on the type of task selected. In the Text Replacements and Translations panel, teachers can set their own means of expression in the desired native language. The latest H5P options of the block allow you to choose the attributes that will be displayed with the task (Download button, Embed button, Copyright button). Thus, the teacher can decide whether the created task will be openly accessible to other learners (either by downloading the task or generating an embedding code) or if references to the sources and licenses used will be published.

In terms of evaluation, H5P has adapted a plugin to capture xAPI instructions. That is, it generates xAPI statements from student interactions. These statements contain information such as: what answers did the students select, what is the score of the question, or whether the question was completed [37].

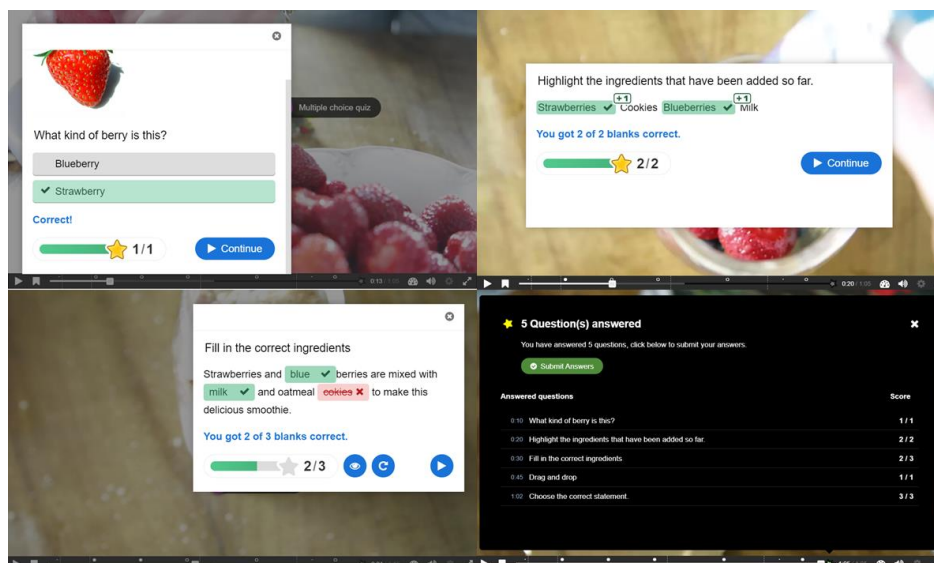


Fig.2.10. An interactive video made in H5P

Among the most featured applications offered by H5P we mention:

- Interactive videos - interactive videos in which you can include questions with several options, to fill in the blanks, the text (see Figure 2.10);
 - Course Presentation - a presentation in which the tutor can also integrate different types of interactions;
 - Branching scenario - a scenario made in the form of a game in which there are various points where the student must choose the right direction to go;
 - Agamotto - allows users to compare and explore a sequence of images interactively;
 - Different types of quizzes - allows tests with multiple types of questions;
 - Virtual tour - a virtual tour content in which can be added questions or texts;
- Interactive content H5P can be seen as a flexible, responsive and versatile mobile tool that promotes an innovative way of linking teaching resources and

multimedia elements. H5P is supported by any LTI-supported VLE such as Canvas, Brightspace, Blackboard and LMS Moodle [37].

2.2. Learning Objects in Learning Management Systems

Once learning management was transferred from the classroom to a new level of development, classroom experiences had to be reconfigured for computerized delivery and distributed over the Internet. E-learning platforms mainly comprise LMSs that focus heavily on creating and standardizing learning content, distributing materials to learners, and providing functionality for self-assessment exercises and examination purposes [8].

A LMS is a software application for the administration, documentation, tracking, reporting, automation and delivery of educational courses, training programs, or learning and development programs [38].

The teacher to create course content by adding text, images, tables, links, slide shows, etc can use LMS. They can manage their courses and modules, enroll students or configure their self-enrollment, as well as import students to their online courses.

In terms of assessment, LMS can allow teachers to create online tests for students. Different types of multiple choice questions are allowed, such as: multiple-choice answer; true or false / yes or no; fill gaps; single-line response; essay and even offline tasks.

Some platforms allow for attendance management and integration with classroom instruction, where administrators can view attendance and records if a student attended, arrived late or did not attend classes.

2.2.1. Learning Objects in Moodle

Moodle is free and open-source learning software, also known as a Course Management System, LMS, or Virtual Learning Space. Since October 2010, there have been 49953 registered users and verified sites and serving 37 million users in 3.7 million courses [9].

Moodle is a learning platform originally developed by Martin Dougiamas to help teachers create online courses focusing on the interaction and collaborative building of educational content, which is constantly developing.

Moodle has customizable management functions. This system is used to create private websites with online courses for educators and trainers to achieve learning objectives. Currently many schools and universities have created their free platform and/or virtual campuses.

The basic functionality of Moodle can be extended with the help of plugins. There are hundreds of variants and any of them can be stored in the Moodle plugin directory.

Any individual Moodle site or course may have different graphics. Themes can be set for the entire site, for each course, but there can also be custom themes that are applied to only one session.

Moodle graphics can be installed to change the look of a Moodle site or an individual course. In Figure 2.11 it is represented a course page for which it was used the Boost theme. Boost is a basic theme that gives Moodle sites a new look. It is easy to configure and offers a better navigation within and between courses. A page using Boost theme is divided into three sections:

- Course sections - is the section in the middle of the page and includes learning materials grouped by weeks, topics, forums or other layouts;
- Navigation drawer - is the block located on the left of the page (Figure 2.11), it is like a map to navigate in course and on the site;
- Gear menu - is the block located on the right of the page (Figure 2.11) and offers different levels of access to teachers and students.

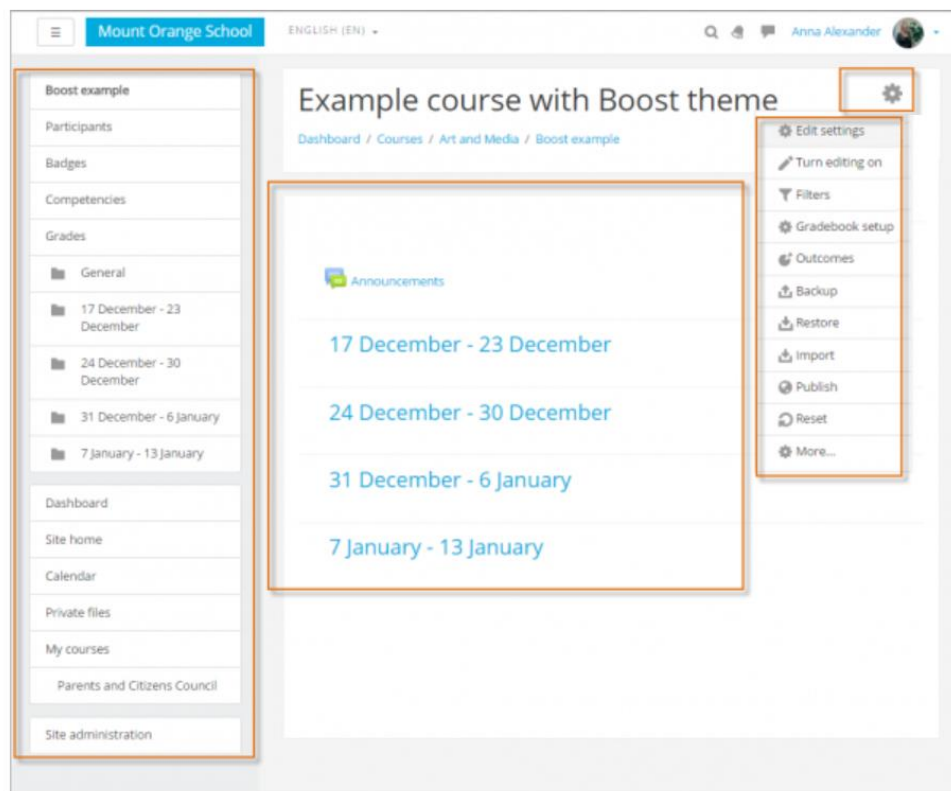


Fig.2.11. A blank Moodle course page using the Boost theme

The themes are based on a responsive web design and allow the use of Moodle on mobile devices. A Moodle mobile app is available on Google Play, the App Store (iOS) and the Windows Phone Store.

Tutors have the opportunity to customize their course by adding resources and activities.

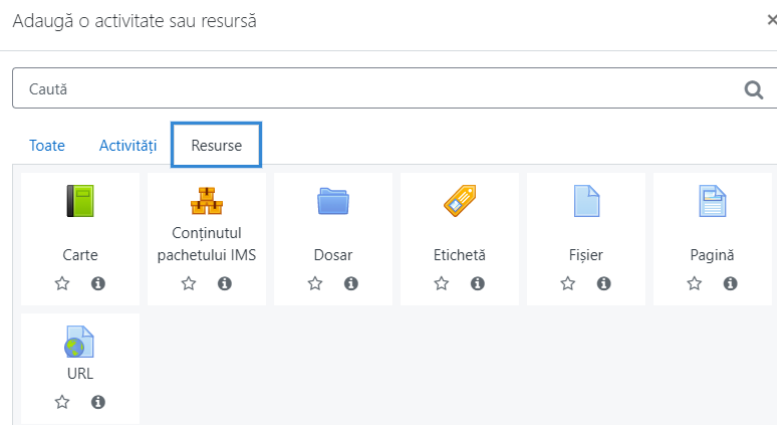


Fig.2.12. Resources provided by Moodle

In terms of resources, several modules can be added (see Figure 2.12):

- Book - this module allows the creation of a resource with several pages, divided into chapters and subchapters like a book;
- File - when you want a file as a course resource;
- Folder - allows you to upload a folder, it is used when you want to display a number of related files in one place;
- IMS content package - allows the attachment of a collection of files packaged according to a standard, it is recommended for the presentation of multimedia and certain content;
- Label - allows the introduction of text and multimedia files in the course page, links to other resources and activities, helps to improve the appearance of the course;
- Page - allows the tutor to create a web page resource using the text editor, it can display text, images, sound, video, web links and embedded code;
- URL - allows the provision of a web link as a course resource, it can be customized and can be incorporated or opened in a new window.

The activities that a tutor can choose on the Moodle platform are:

- Choice - consists of a single question and an offer of possible answers, the answers published on the platform can contain the names of students or anonymously (although teachers always see the names of students and their answers)
- Workshop - allows collecting, reviewing and evaluating answers in a stream, students can send any digital content;
- Database - allows participants to create, maintain and search for a collection of entries, the structure of records is defined by the teacher as a number of fields, and the fields can be check box, radio buttons, drop-down menu, text area, URL, image, uploaded file;
- Chat - allows both students and tutor to have synchronous discussions in real time, in text format;
- Questionnaire - a variety of types of questions can be used;

- Feedback - the teacher can create a personalized survey to collect feedback from students using a variety of types of questions;
- Forum - allows students to have asynchronous discussions;
- Glossary - allows students to create and maintain a list of definitions, such as a dictionary, or to collect and organize resources or information;
- H5P - allows you to upload H5P files and add them to a course;
- External tool - allows students to interact with learning resources and activities on other websites;
- Lesson - allows the teacher to deliver content and / or practice activities in interesting and flexible ways, a lesson can be graded, with the grade recorded in the notebook;
- SCORM package - allows SCORM or AICC packages to be uploaded as a zip file and added to a course, the content can be displayed in a pop-up window, with a table of contents, with navigation buttons, and the notes are recorded in the notebook of notes;
- Attendance - The teacher can create multiple sessions and mark the attendance status as "Present", "Absent", "Delayed", or "Agree", or can change the statuses;
- Task - the teacher can communicate tasks, collect answers and provide grades and feedback;
- Survey - the teacher can use this activity to collect data from students that will help him learn about the class and reflect on his own teaching;
- Test - allows the teacher to create tests comprising questions of different types, including multiple-choice, matching, short answer, and number;
- Wiki - an activity that allows participants to add and edit a collection of web pages can be collaborative, everyone can edit it or individually, where everyone has their own Wiki that only he can edit.

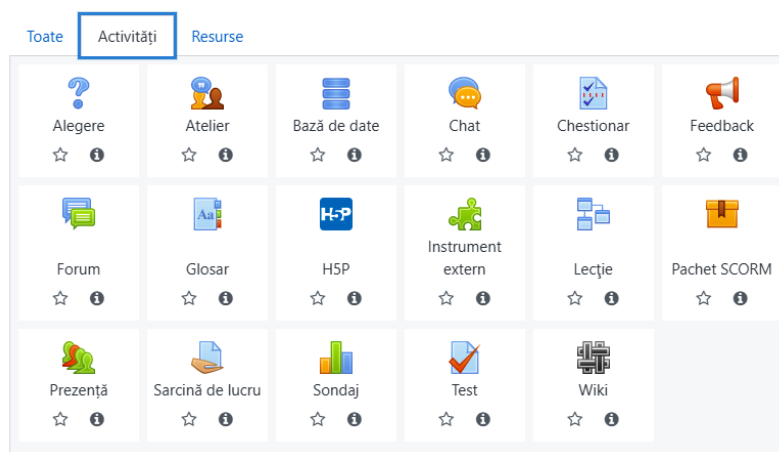


Fig.2.13. The activities offered by Moodle

Moodle Certified Partners offer Moodle services: hosting, training, personalization, and content development [9]. This network of suppliers supports the development of the Moodle project through royalties.

Moodle runs unmodified on Unix, Linux, FreeBSD, Windows, OS X, NetWare, and any other systems that support PHP and a database, including web hosting providers. It also has import features for use with other specific systems, such as importing questionnaires or entire courses from Blackboard or WebCT.

2.2.2. Learning Objects in ILIAS

One of the first LMSs that have been implemented in university education is ILIAS. The prototype was developed in 1997. It was made within the VIRTUS project of the Faculty of Management, Economics and Social Sciences at the University of Cologne. It was initiated and organized by Wolfgang Leidhold [37]. In 1998, the first version of LMS ILIAS was delivered to the Faculty of Business Administration, Economics and Social Sciences in Cologne. ILIAS was published in 2000 as an open-source software, because more universities showed interest in it. A new version called "ILIAS 3" was developed between 2002 and 2004. It became the first open-source LMS to reach the SCORM 1.2 compliance level. Currently, the enhanced version ILIAS 5.3 is used. It is based on HTML5 and can be used as an interoperability provider of the learning tool. The LOs that a tutor can choose on the ILIAS platform are presented below.

A course is a collection of LOs of different types. It can be displayed in different didactical presentation modes. Within a course, learning progress settings can be set. Only course members are able to see the course content.

A group is a container with different LOs. However, unlike courses, groups do not have different didactical presentation modes. Groups are often used to split up course members, like different classes for the same course. Groups can also be used outside the context of a course and are then often used as a collaboration or project tool.

Learning Module ILIAS has an integrated authoring tool that can be used to develop both proprietary ILIAS modules as well as SCORM modules, which can also be imported from other tools like Captivate or Articulate.

Chatrooms can be defined within a course or group to allow communication between members, of between members, and a tutor (private chat).

Survey ILIAS has a survey object which can be used to collect information from several users; for example to evaluate courses or other events. ILIAS surveys are easy to define and manage, even for non-experienced staff.

Personal Desktop - each user has his or her desktop. Here you find all objects that you are registered for, as well as the page markers and notes that you made. You can also collect your own files, documents, websites, and course certificates. You can even develop one or more personal portfolios.

Test & Assessment is an integrated environment for developing and maintaining questions and tests. Tests are used to measure existing knowledge. About a dozen different question types are supported, such as multiple-choice, cloze, drag-and-drop, etc.

Wiki, is a hypertext document that is developed and maintained by some people working together, for example, participants in a course. Wikis are used for virtual collaboration and knowledge sharing.

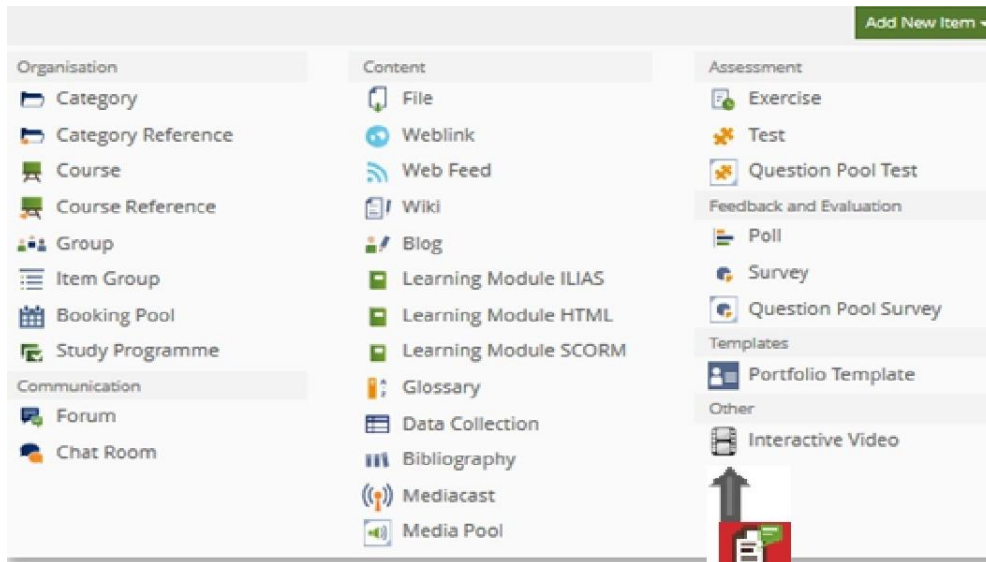


Fig.2.14. LOs in ILIAS

Glossary, is a list of terms with corresponding definitions, much like an encyclopedia. A glossary can be structured like taxonomy and can, as such, serve perfectly as a learning or reference object. Glossaries can also be associated with learning modules (in authoring mode) so that when a term from the glossary is used in the learning module, a hyperlink is automatically created that links to a popup with the corresponding definition. Definitions can include text, but also all different types of media like images and videos.

An organizational Unit, can be created in ILIAS to reflect the organizational structure of the organization. For each organization unit, you can specify supervisors and direct reports.

SCORM is a set of technical standards for e-learning software products to ensure that e-learning products "play well" with other e-learning products. For example, a learning module that is developed using SCORM standards will be played on any LMS that adheres to these standards as well.

A forum or discussion group is an online place where users can discuss issues. Users can communicate asynchronously and a discussion can continue for a long period. In ILIAS, a forum is often used as a means of communication between a course tutor and course participants.

File - files can be uploaded to ILIAS. Files of every type can be uploaded, and subsequently downloaded. However, several types, such as PDF, PNG, MP3, MP4, etc., can be opened directly in ILIAS.

Exercise is a complex learning object. Using an exercise, a tutor can assign one or more tasks to course participants (either to individual participants or to teams of participants). These tasks can have start and end dates. Before the end date, participants need to have finished the assignment and the results uploaded. The tutor will review the submitted results and grade the assignment, also possibly render feedback to the participant. You can also allow participants to review each other's work (peer-to-peer feedback).

A repository contains all learning objects in the ILIAS installation. You can compare it with a catalog or library. The repository is usually organized hierarchically, using categories and courses. A tree view allows for easy navigation through the repository.

A session is a live event within a course. A session has a start and end times, as well as a physical location. Participants are expected to be physically present. Often sessions are used for practice sessions as part of a blended learning approach.

Web Feed is an RSS feed (Real Simple Syndicate). The feed will appear on the side of your screen.

Booking Category is a pool of resources that can be used within a course, like beamers, classrooms, teachers, books, etc.

Item Group is a way to organize objects in the repository and courses.

Data collection is a spreadsheet, like Excel. You define the columns you want to use and enter data.

A poll is a one-question survey. The poll is typically shown on the right side of the screen. A user can only enter the polls once. Results are always anonymous and are available immediately using a pie chart.

A bibliography is a tool to do bibliographical research

A map is an organizational object to create a hierarchy within a course or group.

Web Site is a URL Blog. A blog is short for Web Log. It is a discussion or informational website consisting of separate messages or articles. Often readers can comment on articles and start a discussion that way.

Media cast is a collection of media (videos, images, audio) that are made available to participants in a course.

Media Pool is a collection of media objects that can be used when developing other objects and pages. Within a media pool, you can define a structure using maps.

A portfolio is a personal document in which you describe what you can and how you want to develop yourself further. Its goals are 1) to show your mentor/tutor what you have learned and what you want to do with it, and 2) to help you reflect on what you have achieved and what you want to learn next.

A category is a map that you can use to structure the repository. You can compare it with the folders on your hard disk.

Competency Management is a collection of tools to develop and maintain competencies to develop and maintain function profiles, assigning function profiles to users, creating gap-analysis reports, getting peer-to-peer feedback, and translate this into competencies.

ILIAS provided a flexible environment for learning and working online with integrated tools. It can be seen as a type of library that provides materials and content for learning and working in any location of the warehouse. There are many universities that use ILIAS, an example is the University of Bonn, see Figure 2.15.

The screenshot shows the eCampus ILIAS page for students. At the top left is the University of Bonn logo and the text 'UNIVERSITÄT BONN eCampus'. A search bar on the top right contains the text 'Website durchsuchen'. Below the header is a navigation menu with items: 'eLearning services', 'Design teaching digitally', 'Events', 'Instructions & links', 'For students', and 'about us'. On the left side, there is a vertical menu with links: 'Online learning modules', 'Videos and recordings', 'eCampus learning platform', 'e-exams', 'Events for students', 'Courses from the University of Cologne', and 'FAQs'. The main content area starts with the breadcrumb 'You are here: Home → For students' followed by the heading 'eLearning for students'. Below this is a photograph of students working at laptops in a library setting, with the caption 'Foto: Frank Homann / Universität Bonn'. To the right of the photo is a login box titled 'eCampus Lernplattform LOGIN' with a key icon and a password field. Below the login box is the 'eCampus support hotline' information: 'Mon-Fri | 10:00 a.m. - 4:00 p.m.', 'Tel: +49 (0) 228 73-5092', 'eMail: ecampus@uni-bonn.de', and a 'Contact form' link. At the bottom of the page, there is a paragraph of text: 'You can find your courses in digital format on eCampus. Lecturers post information and materials here and communicate with their students. Various tools can be used for digital teaching. You can discuss and work together in forums, blogs and wikis. Knowledge and competencies can be developed and checked in tests and learning modules. With eLectures, teachers can provide introductions and lectures in video format. Discover the possibilities of eCampus and digital teaching together with your teachers!'

Fig.2.15. An e-campus ILIAS page

ILIAS provides reports on learning progress both in individual LOs and in whole courses or groups. They show the processing state of a student, his progress in terms of learning objects. This facility does not need access to the administration area and it is available directly to the course.

LMS ILIAS is open source, powerful and flexible. It can be used for an online learning and testing platform for schoolchildren and students, a training environment for employees, or an e-learning library.

2.2.3. Learning Objects in Blackboard Learn

Blackboard is one of the most popular names in the digital learning market. The platform comes in both software-as-a-service (SaaS) and non-SaaS models [39]. The service provider offers all core learning management features as well as powerful data analytics, communication channels, collaboration tools, and web conferencing. Class facilitators can easily deliver homework, tests, and track grades. They can also manage online and blended classes.

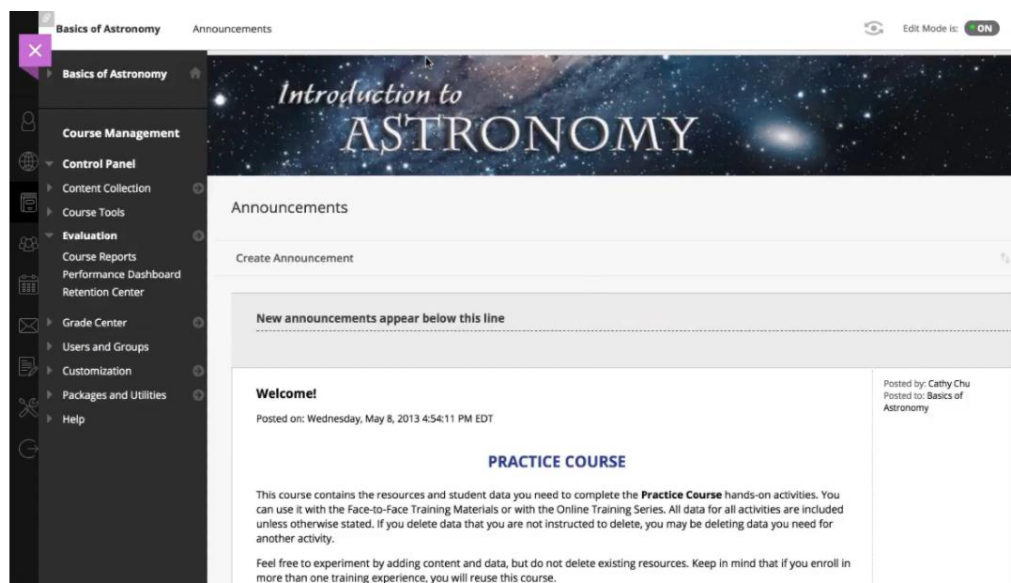


Fig.2.16. A course on Blackboard Learn

Blackboard Learner, has a responsive user interface for improved accessibility. In addition to the basic functions specific to an LMS, it also comes with Blackboard Analytics. Blackboard Analytics allows tutors to see valuable information, such as student performance, involvement, and more. The manufacturing company has also launched the Blackboard Open LMS powered by Moodle. This is a more extensible version of Blackboard services, where you can customize various aspects of some courses.

Mainly, Blackboard targets enterprise and mid-level market segments. They provide specific customer services. Customers who want Blackboard must book a demo in order to get a personal offer.

Blackboard Unite for K-12 is a comprehensive digital learning environment that enables personalized learning anytime, anywhere. Blackboard Learner has a modern, intuitive and responsive interface. It offers a simpler, more powerful teaching and learning experience that goes beyond the traditional LMS. Blackboard, is more than an LMS, it is a model in which the emphasis is not on individual products. It is an integrated and flexible ed-tech platform. It that offers a uniquely connected experience in a wide range of capabilities designed to stimulate student success and institutional performance.

The ed-tech platform offers three essential components:

- a design and architecture approach that allows for a complete user experience in a broad set of capabilities that will continue to evolve;
- the ability to model the digital environment in the way that best suits the requirements of the beneficiary;
- the power to enable better outcomes for institutions, instructors, and students.

This approach to building, delivering and expanding Blackboard solutions creates a very popular connected and extensible educational technology ecosystem.

2.3. Generative Learning Objects

2.3.1. GLOs Coined by Tom Boyle

Improving teaching and learning is increasingly achievable with ICT. However, to get the maximum impact it needs a "stratified learning design" [25]. This approach allows tutors to create different levels of abstraction, reaching the design of activities focused on specific learning objectives.

"Learning objects were often monolithic" [14], so there was a need for them to develop to be able to respond to issues such as reusable, repurposable, decoupled, pedagogically rich, available for independent reuse. These challenges have led to massive work that has materialized for Boyle by building LOs in the study of Java programming. This approach won, in 2004, the European Academic Software Award (EASA).

The concept of a GLO has as a basic idea the realization of a successful LO with the ability to reuse, not necessarily focusing on the content.

Respecting the principles of software engineering in generating LOs, Boyle found it as the first challenge to get the simplest and most decoupled object possible. The "principle of de-coupling" as it is called in [14] implies that each object must have minimal connections with the other units. Thus, each object of learning must relate to its material and not be dependent on others.

Boyle said: " We must face the challenge of creating LOs that are cohesive, decoupled and pedagogically rich" [14].

Another problem ignored in the main approach of standards-based learning objects is the design of high-quality learning objects. Boyle argued that "high-quality design and development of learning objects is crucial before we get to issues of metadata and software packaging" [16].

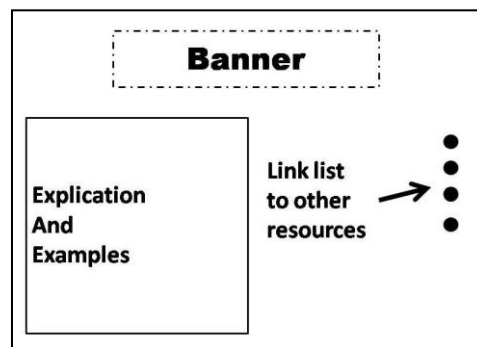


Fig.2.17. Schematic layout of the format for LO realization

He was focused from the beginning on high-quality design. The structure of Boyle's design for layouts is simple and very flexible [21]. Each object is made in the form of a web page composed of two parts:

- one consisting of the text integrating the explanation;
- one consisting of a link column that offers associations with other objects.

To make an efficient model, Boyle said: "learning resource ... should be designed with reuse in mind" [40]. He used for this purpose the reuse of design, more precisely a method of reuse design known as patterns. Models are similar to classes, but they have a wide scope and may include more classes.

Another tackled issue of repurposing without accessing the skills of a specialist e.g. multimedia developer. The model is based on patterns that can be easily used.

A different tackled issue for GLOs is one of being available for independent reuse. Instantiations are available by the instrumentality of a web browser and can be used for individual study.

Another challenge was to make these designs accessible to tutors through a tool that permits the creation and adaptation of LOs.



Fig.2.18. Representation of functional choices in GLO-Maker

To achieve this goal a tool has been created to incorporate GLO models and make them available both for creating new learning objects and for adapting existing objects for studying. Boyle performed a special tool, called GLO-Maker [16]. That tool was developed to make it possible for the users to create based on design and adapt as needed generative learning objects. The first version was created in 2008, having the interface is shown in Figure 2.18 [16].

Tom Boyle's project turned into a resource creation tool called GLO Maker [41]. It is open-source and free for educational use. The GLO-Maker tool provides two main interfaces: one allows the user to access a pedagogical design expressed as a structured set of pedagogical choices the 'surface structure' (see Figure 2.19) [16] and one that expresses these choices as a sequence of screen layouts.

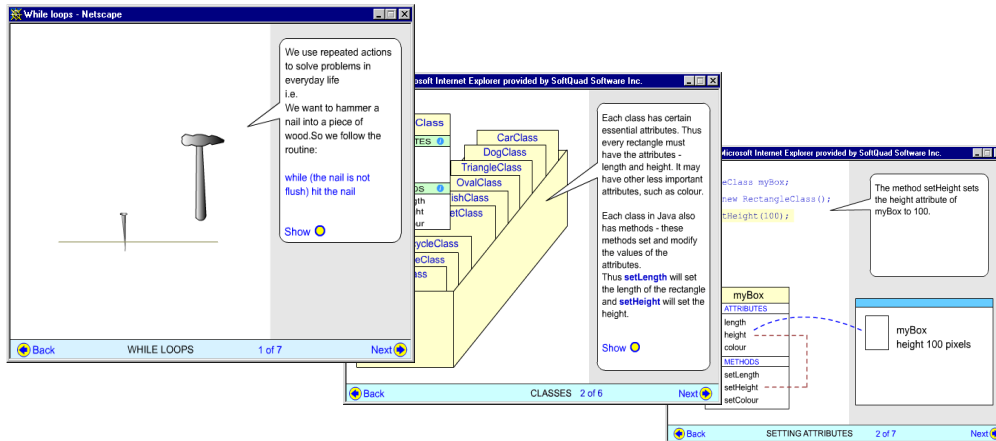


Fig.2.19. Examples of the 'surface' structure 'pages' of GLOs as seen by learners

Each learning object is saved in a separate package in its directory and can be played from any location into which this package is moved.

Each learning object should be based on one clear learning objective. The tutor can thus handle students differently, depending on the level of knowledge they have achieved. Some learning objects are just to practice simple notions while others are using advanced questions.

Boyle approached these problems using a form of representation borrowed and adapted from Generative Linguistics, especially Systemic Grammar [42]. The final form of an object and the description of the processes involved in its production can be radically different. Systemic Grammar is used as the basis for the GLO approach because [14] provides the basis for a form of representation that is both amenable to understanding by human users and executable by formal computer software" [16].

The process of developing a GLO involves several stages:

- in the first step a series of screen layouts or templates were designed by tutors and sometimes learners;
- in the second step, there were extracted and represented formally the pedagogical decisions underlying the generation of the learning object and
- in the third step, the templates were captured in a network.

The central structure is a selected and organized set of pedagogical choices. It is captured as a decision structure - in which each node represents a pedagogical function that can then be refined or expanded as needed.

Learning objects are not scalable. The concept of generative learning objects (GLOs) was developed to tackle the problem of facilitating scale-up [40]. The tool realized by Boyle has the property to be scalable. The tutors can use it to make their learning resource, they can add pictures, audio files, and whatever they think is useful.

The organization of concepts is essential to have a stable and efficient use of technology in the learning process. Technology is appealing, but it can distract attention from "profound design" [43]. It requires a base, a prototype, for design, and a conceptual unit to apply and keep up with technological change.

2.3.2. GLOs in the Works of Damasevicius and Štuikys

The model proposed by Vytautas Štuikys and Robertas Damaševičius is reusable and is quite general, can easily be customized to the tutor's need. The model is expanded with meta-programming technologies. Their approach aims to extend the existing GLO concept, their content relying on pedagogical, social, and technological aspects [18].

They have developed a model that responds to the problem of decouples, including lessons in mathematics, software engineering, or computer science domain. They have proposed to focus on the instantiations of learning objects. The model is represented in Figure 2.20.

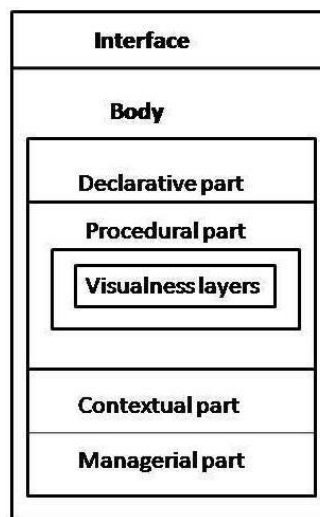


Fig.2.20. The structure of the GLO model

This model is composed of three parts:

- name, for identification;
- interface, which transports information from LO and to LO;
- body, which is a list of accumulated learning information.

The model proposed by Robertas Damaševičius and Vytautas Štuikys also responds to the problem of reusability. In [13] is an example of GLO for array sorting algorithms. The processor generates a learning object implemented in two parts: one is represented in HTML and contains the description of a sorting algorithm and the second part is in Javascript and demonstrates the efficiency and principles of the algorithm. This GLO can be distributed over the internet and can be viewed on both a computer and a phone (see Figure 2.21).

As a result, they obtain a GLO expanded known LO model with the help of meta-programming techniques for generating instances. This model can be used to teach Computer Science at the university or high school level.

In [44] the authors reused the model using LEGO robots. In this variant, there is present both educational and technological sides. The LEGO NXT robot has been easily introduced into this area. It is used to demonstrate the functionality of programs and algorithms. This combination was possible because the models are independent

of the instruments and tools used. The robot-based approach also has the effect of increasing external variability and improving internal reuse.

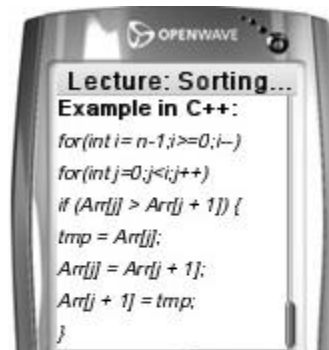


Fig.2.21. Example of LO: Bubble sort; C++; on Mobile (a fragment)

In the latest works, there is a real interest in the research field on knowledge transfer “from the teacher’s ontology to the learner ontology” [45].

Stuikys starts with the GLOs presented in [18] but together with other collaborators, they had extended a set of GLOs to a specific meta-programming template called the stage-based GLO model [46]. This model allows the implementation of a more flexible approach and the automatic adaptation of content through stage-based transformation.

This direction has been chosen as more efficient solutions are sought in terms of harmonizing the pedagogical and technological side. Technological standardization is achieved when content is explicitly specified in stages and then interpreted using an appropriate automation tool. Thus, the step-based model is what is achieved at the meeting between the pedagogical stage and technological standardization.

This model differs from the first variant of GLO by the following:

- the internal structure is multi-stage;
- content generation is phased;
- the process of generating is influenced by context.

In this case, the context is understood as any information that can be used to characterize any relevant interaction between the user and the application [46].

2.3.3. Moodle Coordinate Questions

Moodle Coordinate Questions (MCQ) are a plug-in for Moodle [9]. Through these questions, the student can answer a set of correlated and random calculation questions. They are mainly used in physics and engineering. The main idea is based on the fact that the calculation of numerical (random) data can be turned into answers and can be verified.

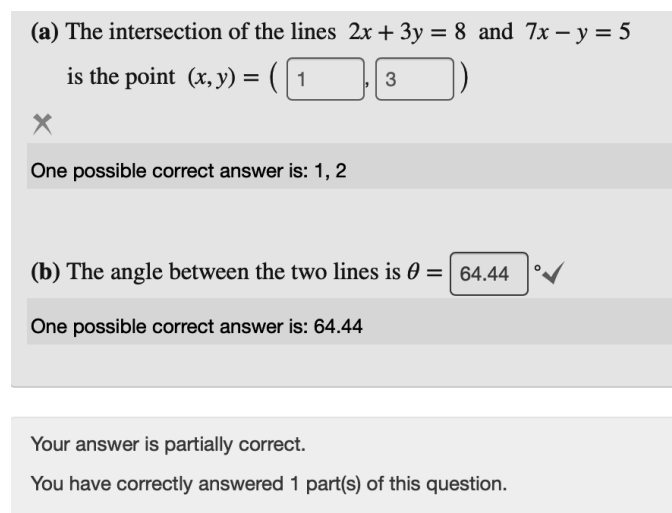
These questions are meant to be generic so that different types of questions can be easily created.

The main features of these types of questions are:

- based on the choice of random numbers from a set, each student can receive an individualized question;
- the answers can be of four types: number, numeric (number and arithmetic operation), numerical formula, and algebraic formula;

- multiple sub-questions can be performed based on the same random variables. A sub-question is valid if it is assigned a grade and an answer;
- it is possible to specify grading criteria to verify the correctness of a student's answer. An answer is not only wrong or correct; it can also have intermediate notations;
- it is possible that a question can be accessed multiple times, but in this case, a maximum scoring sequence needs to be specified;
- the correctness of the answer is checked and graded so that the student automatically sees his mark.

The tutor can edit the text of a question. In addition to text, variables and substitutes can also be used. The names of the variables will be replaced by the numeric value in the version that the student receives.



(a) The intersection of the lines $2x + 3y = 8$ and $7x - y = 5$ is the point $(x, y) = (\text{input box: 1} , \text{input box: 3})$

✘

One possible correct answer is: 1, 2

(b) The angle between the two lines is $\theta = \text{input box: 64.44}^\circ$ ✓

One possible correct answer is: 64.44

Your answer is partially correct.
You have correctly answered 1 part(s) of this question.

Fig.2.22. Example of Moodle Coordinate Question

The drawback of these kinds of questions is that they can be applied only where it is algebraically responsive, as can be seen in the example of Figure 2.22.

Structurally, the MCQ has four sections: main question, sub-question, extra-options, and variables instantiation checking.

The main question section describes the context of the exercise. It also includes values of the random variables expressed by enumeration having a step that is default 1, but that can be changed.

The sub-questions section contains the text, the unit and the grading criteria. These sub-questions tend to reuse the set of random variables instantiated in the main question, giving the tutor the opportunity to further develop their exercise ideas.

The extra options section contains a set of specific options that apply to all sub-questions.

The variable instantiation check section is a section in which some questions are displayed to the tutor, optionally with answers, so that he has a visual representation of the exercises that the student will see [47].

Depending on the type of answer, it can be either an evaluated expression or a string. If the answer is a number, then an error variable is calculated as the

difference between the calculated answer and the given answer. The answer is correct if the difference is less than a defined threshold denoted absolute error.

2.3.4. Moodle Calculated Questions

Moodle Calculated Questions are a plug-in for Moodle [9]. Calculated questions offer the ability to create multiple versions of a question with different numerical values. Metacharacters are used to ask such questions. A metacharacter is obtained from an ordinary variable enclosed in parentheses, for example the metacharacter {a} can be created. These metacharacters are substituted with certain values during the test.

When Moodle delivers a calculated question to the student, the metacharacters are replaced with randomly selected values from a predefined set of possible values. This allows control over the possible values so that the numbers are realistic.

There is two types of data sets:

- private, used by a single calculated question;
- shared, used in all calculated questions that use it.

When Moodle delivers a calculated question to the student, the wildcards are replaced with randomly-selected values. However, these values are not completely random - rather, they are randomly selected from a pre-defined dataset of possible values. This allows you some control over the possible values chosen - for example, in order to make sure the numbers are realistic.

These datasets can be private or shared - private datasets are used by one wildcard within one calculated question; shared datasets are used by one wildcard within all calculated questions that use it.

To create or modify a calculated question there are three pages to work through.

In the first page, we edit the calculated question. Any shared wildcards for this category are listed beneath. If we change category, we will need to click the "Update the category" button to refresh this list. There may not be any shared wildcards yet - if not, we can create them later if we wish. Next, we add the formula for the answer. This formula must contain at least the wildcards that appear in the question text. We can see the correct answer formula syntax for further details. Choose the grade that the student will get for this question if they give this answer. This should be a percentage of the total marks available. For example, we could give 100% for a correct answer, and 50% for an answer that is nearly right. One of the answers must have a 100% grade. Determine the tolerance for error that we will accept in the answer. The tolerance and tolerance type settings combine to give a range of acceptable scores. So, if tolerance = t , correct answer = x and the difference between the user's answer and the correct answer is dx .

In the second page, we chose dataset proprieties. If there is anything in the question text that looks like a wildcard, but does not appear in any of the answer formulae, we can specify whether or not this is meant to be a wildcard. If it is, we can choose whether it should use a private or shared dataset.

In the third page, we edit the dataset. Now we need to create the set of possible values that each wildcard can take. There are two ways of creating values - you can type them in yourself and add them to the list, or you can have Moodle generate them for you.

Grade 9.60 out of 10.00 (96%)

Question 1
Partially correct
Mark 1.60 out of 2.00
Flag question
Edit question

Calculate the area of a square with side 18 m.

Answer: ✓

You did not give the correct unit.
the area is the square of the side length
The correct answer is: 324 square meters

Question 2
Correct
Mark 5.00 out of 5.00
Flag question
Edit question

Calculate the area of a rectangle with sides of 2 and 46 m, respectively.

Answer: ✓ m

The area is the product of the two measures.
The correct answer is: 92 m

Question 3
Correct
Mark 3.00 out of 3.00
Flag question
Edit question

The square is

- a. trapezoid with an angle of 4 degrees
- b. the parallelogram with 94 congruent sides
- c. the rectangle with two consecutive congruent sides
- d. the rhombus at a 90 degrees angle

Fig.2.23. Moodle Calculated Question example

In Figure 2.23 an example of a test consisting of three questions is presented, the first two of which are calculated questions. The first question has the partially correct answer, and the second has the correct answer. In both cases, the student receives feedback. If the answer is incomplete, it is warned about what is missing. If the answer is correct, he is reminded of the theory that is applied in the calculation. In addition, in both cases the correct answer is presented. Calculated questions allows only arithmetic operators and functions, possibly in addition a unit of measure.

2.4. Learning Objects in Other Works

2.4.1. Learning Objects Sequencing Papers

Course materials consist of reusable LOs grouped in sequences. LO grouping can be accomplished through different processes by the tutor or automatically. A model that can perform the sequencing is presented in [48].

Web-based courses and adaptive systems can replace the tutor and adapt to different environments. These systems can use a variety of educational methods. One of these methods is the sequencing of the curriculum designed to give students individualized material [49].

In the case of classical learning objects, the sequencing is done by the tutor, the course thus created can't be personalized, on the contrary, it is general. It is therefore desirable to create a model that can automatically perform sequencing.

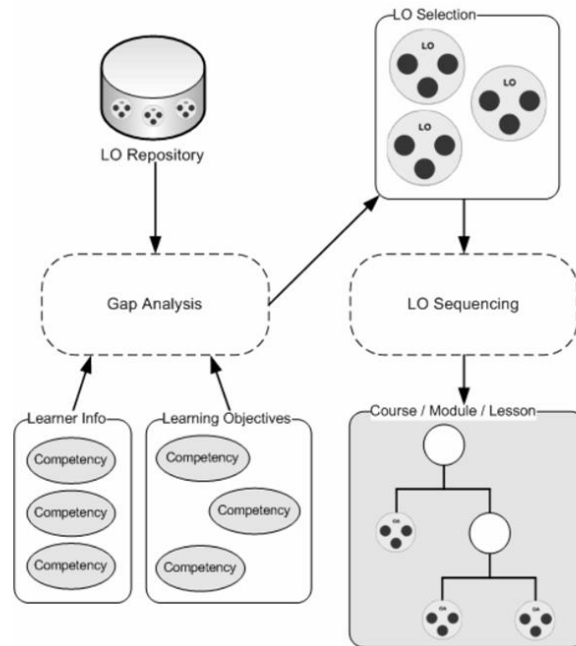


Fig.2.24. The LO sequencing process

Thus sequencing is one of the solutions proposed by Luis de-Marcos and his collaborators in [48] to solve the problem of flexibility and to automate the role of tutor. What they have proposed uses artificial intelligence techniques. Learning objects are seen as small reusable learning units that can be grouped to get a lesson or a course.

The model in [48] is defined for the LO sequencing according to the competencies they are targeting. This involves attaching competencies to each LO. A valid sequence of LOs corresponds to a set of competencies.

Modeling of students is done according to competences. Those are also used to outline the expected learning outcomes. The process of analyzing the existing gaps in knowledge and the distribution of LOs aimed at exactly filling the identified gaps will be performed through the squinting process (see Figure 2.24).

The realization of these associations is done through a classic problem of artificial intelligence, namely the problem of constraint satisfaction. A Progressive Swarm Optimization (PSO) algorithm was used to optimize the sequencing. Thus the problem of sequencing is transformed into a permutation problem. The authors' experiments show that PSO manages to solve the problem they were considering.

The sequencing process was actually tested in engineering classes as well as in the laboratory. The authors have shown that this process works even in complex scenarios in which people face difficulties. This will save time and reduce costs. The model can also be extended to build personalized e-learning experiences. The sequencing process can be completed with a gap analysis process. The courses created by this method can be integrated into LMS platforms.

2.4.2. Learning Objects Improved by Object-Oriented Design

Claudine Allen and Mugisa Ezra consider it necessary to have a consistent LO theory since the very vague definition of an LO has led to the absence of a well-defined theory and a variety of object interpretations.

In [50], several LO standards are analyzed to identify areas where improvements could be considered. As problems of the LOs theory are emphasized in the paper: level of granularity, incompatibility between models, LO reusability. Under these circumstances, it is considered useful to have a standard that combines LOs of any size. The authors present their theory about the existence of a standard for LOs in the article.

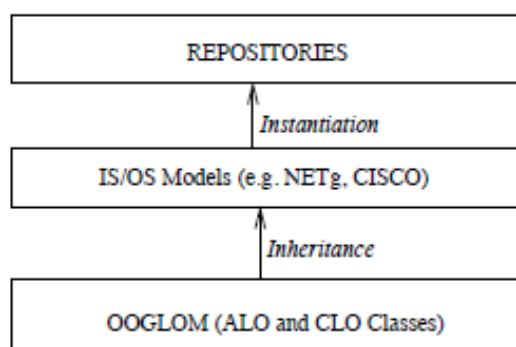


Fig.2.25. OOGLOM as a base for IS/OS Models

The proposed theory defines seven concepts:

- Learning Experience is a series of events that lead to learning goals;
- Learning Objective is an assertion that includes measurable knowledge, abilities, or attitudes that students have to gain through learning experiences;

- Learning object independence refers to the ownership of an LO to be self-sustaining from a pedagogical point of view, but not necessarily from a technical point of view;
- E-Learning refers to the use of ICT in learning experiences;
- Pedagogical activities are educational activities such as assessment activities, instructional activities, curricular activities, or research activities;
- Learning components is material made by an LO to achieve a goal. They run on specific e-learning systems.
- Institution Specific or Organization Specific Models (IS/OS Models) are the e-learning standards that exist.

Object orientation is regarded to be an important factor in solving identified problems as efficient features such as inheritance, polymorphism and instantiation could be used. Thus an LO is an instance of a class. In the proposed model there are two types of LO: atomic learning objects (ALO) and composite learning objects (CLO).

ALO is a subclass of the LO class and is made up of content, metadata, a reference to a learning objective, and data handling and use operations. CLO is a composite LO class instance that consists of a list of ALOs or CLOs, metadata, and data handling and usage operations. A CLO can cover more specific LOs.

The LO theory proposed in [30] includes their model called "Object Oriented Generic Learning Object Model"(OOGLOM). Because it is desirable to eliminate the limitation of reusability and interoperability, there is no working definition of an LO. The authors claim that LOs made according to this theory have the following properties: independence, granularity, reusability, assemblability, contextuality, interoperability, and flexibility.

The OOGLOM model can be represented as a LO tree in which the leaves are ALO, and when running the RunLO main operation of a CLO will recursively call all RunLO operations of all LOs included.

OOGLOM is presented as a powerful platform that provides common base classes from which any other LO model could be developed. It is presented that OOGLOM's popular standards build on the flexibility, reuse, and interoperability of LO.

In [30] OOGLOM is considered useful to be used as the basis for IS/OS Models. The repository of these models can be populated by creating LO class instances. This is illustrated in Figure 2.25.

2.4.3. Learning Objects in Cloud Frameworks

Computational Science is a field in which changes occur very quickly due to research and technical evolution. To meet this challenge in [51], the authors have proposed a hybrid learning resource approach with support for learning objects in cloud computing platforms.

They identified two major trends in modern education at the university level: individualization and mass character. ITC entered all branches of science and offers access to distance learning. It has facilitated the creation of learning objects that support individualized learning. Reusable learning objects replace traditional forms of learning, especially in this dynamic field, such as Computer Science.

However, there is an inconvenience, the preparation of learning resources long, so the rapid integration of research into practice is deficient. Effective practical

work experiences are required in Computational Science due to the rapid growth of research in this field.

The approach from [51] contains hybrid learning resources (HLR) which contain different objects and resources:

- informational object (IO) – a theoretical item,
- informational resource (IR) – some IOs and links to IOs all associated by a property,
- learning object – fuses learning with assessment tools,
- abstract learning resources (ALR) - contains information about how to use efficiently IRs and LOs.

An ALR together with the technical and educational context is a hybrid learning resource. ALR frames are made up of research objects linked together.

In Computational Science, the form of representation of the results underlying HLR research is the software package. A software package can be considered as a function between input and output data and is appointed with all documentation: notations, ways of execution, description of both functionalities, and format of the input and output data. The software package is mapped and represented as a cloud package.

In [51] the authors used the CLAVIRE e-Science platform as the basis for HLR implementation, which can be developed in a relatively short time. CLAVIRE (CLOUD Applications VIRTUAL Environment) is a platform whose structure and functionality are presented in [52].

To incorporate a software package into CLAVIRE, the Easy Package was used. Creating and using cloud and scientific workflow (SWF) packages in CLAVIRE is done with the Domain-Oriented Interface and SWF editor. In this case, a student can both create an SWF from a given package and experience some existing ones. The platform gives the advantage of reaching articles and scientific papers.

CLAVIRE environment is based on intelligent problem solving environment (iPSE) concept. So it is been used as simulation environment for Virtual Simulation Objects (VSO) concept [53].

CLAVIRE is been used as simulation environment. In [53] a use case of using the platform in a simulation environment is presented.

Virtual Simulation Objects is a concept which forms theoretical basis for building tools and framework that is developed for system-level simulations using existing software modules available within cyber-infrastructure.

To put the presented concept into practice, the implementation of VSO management system software is now being developed. It is integrated with CLAVIRE platform which allows building composite the applications with the use of the set of the domain-specific software available within the service-oriented distributed computational environment. CLAVIRE environment is based on intelligent problem solving environment (iPSE) concept. So it is been used as simulation environment for Virtual Simulation Objects (VSO) concept.

To test the implemented system the ship behavior during sailing in the sea was simulated.

Simulation is run using CLAVIRE environment which is provided with AWF with blocks describing corresponding software running. CLAVIRE user interface, which is web-application, is presented at Figure 2.26.

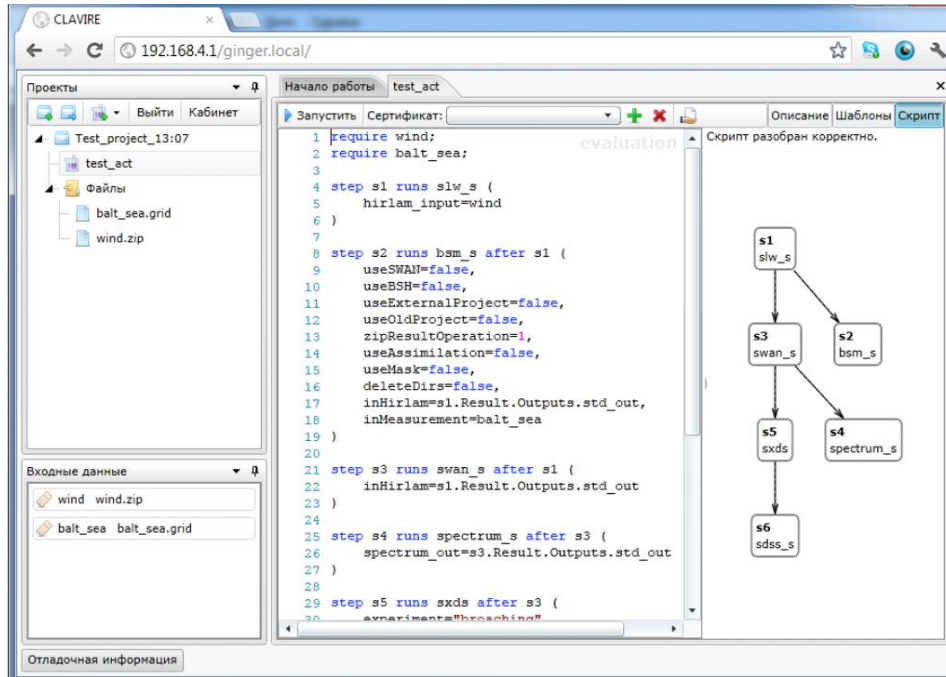


Fig.2.26. Example of LO in CLAVIRE

Thus, this approach allows the use of simple components such as IOs and IRs to a high level of abstraction. This fact also supports the reusability of the packets and quick response in case of changing the input data. This model strengthens the cloud platform both as a packet location and as a learning environment, providing access to diverse teaching materials. Students can even participate actively in the process of creating knowledge by creating their own models and sharing them with others.

A study has been made in the High-Performance Computing Department of ITMO University. The results of the study based on the implementation of the model in [51] confirmed that HLRs are a deep approach than traditional learning.

2.4.4. Learning Objects Records - Experience API

The realization of the Experience API (xAPI) project was possible thanks to the Experience API Working Group, which enjoyed the support of the Office of the Deputy Assistant Secretary of Defense, and the Advanced Distributed Learning (ADL) Initiative.

The ADL has the role to manage the development of the xAPI. ADL believes that xAPI is "an evolved version of SCORM that can support similar usage cases but can also support many of the use cases gathered by ADL and submitted by those involved in distributed learning that SCORM could not enable" [24]. So, the improvements have been implemented based on feedback from active users. From this point of view, it can be said that the Experience API (xAPI) is SCORM's successor.

They also believe that xAPI is among the first technologies that enable a larger architectural vision of online learning and training. xAPI is committed to supporting modern and needed technologies such as authentication services, visualization services, personal data services, querying services.

The version includes also some refinements, such as adding attachments, using JSON rather than XML for Activity metadata storage, signing statements, the name of the document is APIs. It has been designed to support new features such as: translating across platforms, using e-learning on mobile, greater content control, tracking real-time performance, team-based e-learning.

It is a way that facilitates learning anytime and anywhere. This standard can be implemented on different systems and is intended to be used to develop interoperable tools, systems, and services that are independent and communicate with each other. "All learning events are stored as Statements" [24]. Statements represent the xAPI kernel. Statements are learning events. A statement resembles a sentence like, "I did that."

As a distinctive note, the xAPI specifically offers the following outfits:

- structure and definition for how the experiences are communicated by an Activity Provider (namely Statement, State, Actor, Activity, and Objects);
- data transfer methods of objects to/from a repository in terms of storage and data recovery;
- security methods for the exchange of information with safe sources.

An Activity Provider is a software object that communicates with LRS to store information about the experience. The means are:

- Actor – an identity of a person or group being watched using statements as doing an action in a learning activity, is the "I" in "I Did This";
- Activity – interpretation of Activity is broad, it could be a training unit, an experience or a performance;
- Objects – is the "This" in "I Did This";
- Statement – a simple set of <actor><verb><object>, with <result> in <context> to target an aspect of a learning experience;
- Verb – an action in an Activity, is the "Did" in "I Did This".

Actor, Verb, and Object cannot be missed, the other properties are optional. The statement is invariable. Activities are not considered to be part of the Statement and are variable.

If the Actor is a group, two options are available: anonymous and identified. An anonymous Group describes a group of people that does not have an identifier. An identified Group contains exactly a unique Inverse Functional Identifier, which includes a value that is guaranteed to always be used only to identify that group.

The Verb in an xAPI Statement describes the action between the Actor and the Activity during the learning experience and specifies a meaning that is not related to any particular language. The model defines how verbs are created, but does not provide a predefined list of them, as it may not effectively cover all possible learning experiences. The Verb type is IRI (Internationalized Resource Identifiers), is human-readable, and involves the Verb meaning. It also includes a set of names corresponding to languages or dialects.

```

    "actor" : {
      "objectType": "Agent",
      "name" : "Example Admin",
      "mbox" : "mailto:admin@example.adlnet.gov"
    },
    "verb" : {
      "id": "http://adlnet.gov/expapi/verbs/voided",
      "display": {
        "en-US": "voided"
      }
    },
    "object" : {
      "objectType": "StatementRef",
      "id" : "e05fa483-acaf-40ad-bf54-02a8ce485fb0"
    }
  }

```

Fig.2.27. A Statement that voids a previous Statement

There is only one reserved verb called "voided". One of the key factors of xAPI distribution is that it is guaranteed that Statements cannot be changed or logically deleted. Mistakes may happen, and for such situations, there is a need for a Statement to be declared invalid. So a previously made Statement is marked as invalid through "voiding a Statement". Any Statement that cancels another Statement cannot be annulled itself. In Figure 2.27 is illustrated an example Statement that voids a previous Statement which it identifies with the id "e05fa483-acaf-40ad-bf54-02a8ce485fb0".

```

    "id": "fd41c918-b88b-4b20-a0a5-a4c32391aaa0",
    "actor": {
      "objectType": "Agent",
      "name": "Project Tin Can API",
      "mbox": "mailto:user@example.com"
    },
    "verb": {
      "id": "http://adlnet.gov/expapi/verbs/created",
      "display": {
        "en-US": "created"
      }
    },
    "object": {
      "id": "http://example.adlnet.gov/xapi/example/simplestatement",
      "definition": {
        "name": {
          "en-US": "simple statement"
        },
        "description": {
          "en-US": "A simple Experience API statement. Note that the LRS
(learner), the verb, or the activity/object."

```

Fig.2.28. Example of a simple statement

xAPI is a service that allows learning experiences to be delivered and stored in a repository appointed Learning Record Store (LRS) [24].

Each identifier used for any means is unique, since using an identifier in two different cases would cause ambiguities in the Statements' validity. The simplest Statement uses all the necessary properties (Figure 2.28).

You can use references to statements that already exist, they are in the form of pointers and have the name Statement Reference. For example, if there is a Statement with the ID 8f87ccde-bb56-4c2e-ab83-44982ef22df0, then using a new statement we can make another Statement that refers to the original; this is illustrated in Figure 2.29. Sub-statements may also be used, and are included as part of a parent statement. These are done using the "SubStatement" type for the "objectType" property.

There are also optional fields related to a Statement like the "context" field. This field can store information such as a tutor for an activity if this experience has occurred in teamwork or as an individual experience if it is a sub-activity in a broader activity. Four types of contexts are valid:

- Parent - an Activity with a direct link to the Object of the Statement. For example, A test question will have parental activity, the test;
- Grouping - an Activity with an indirect link to the Object of the Statement. For example a C ++ course within a programmer's qualification, course is composed of several lessons and is referred to as a parent for them, but the qualification relates to the lessons as the grouping;
- Category - an Activity used to divide into categories the Statement. For example: if a student tries an algebra test and uses the MTI-3 profile, then Activity refers to the test, and the category is the MTI-3 profile;
- Other - an Activity that does not fit in any of the above categories. For example, a student studies a textbook for a math exam. The Activity refers to the textbook, and the exam is a context Activity of type "other".

```

"actor" : {
  "objectType": "Agent",
  "mbox": "mailto:test@example.com"
},
"verb" : {
  "id": "http://example.com/commented",
  "display": {
    "en-US": "commented"
  }
},
"object" : {
  "objectType": "StatementRef",
  "id": "8f87ccde-bb56-4c2e-ab83-44982ef22df0"
},
"result" : {
  "response" : "Wow, nice work!"
}

```

Fig.2.29. Statement reference example

For a better understanding of these, we can take the following hierarchical structure as an example: "Questions 1-5" are part of "Test 1", "Questions 6-10" are part of "Test 2". These tests, in turn, are part of the "Graphs" course. The first five questions are recorded as part of the test by declaring "test 1" as their parent, and the next five are recorded as part of the test by declaring "test 2" as their parent. They are also grouped with other "Graphs" statements to fully reflect the hierarchy. This is particularly useful when the object of the statement is an Agent, not an Activity. "Tutor X mentored student A with context Graphs." The example is illustrated in Figure 2.30.

```
{
  "parent" : [{
    "id" : "http://example.adlnet.gov/xapi/example/test1"
  }]
  "grouping" : [{
    "id" : "http://example.adlnet.gov/xapi/example/Graphs"
  }]
}
```

Fig.2.30. Example of the "context" field

The authenticity and integrity of a statement are ensured by the fact that a Statement may contain a digital signature. In order to include the original serialization together with the signature, signed Statements include a JSON web signature as an attachment.

Regarding the way the statements are transferred between LRS and the provider, it is not necessary for tutors to fully understand all these details. Some libraries that manage these activities are still being developed for more technologies (such as JavaScript). ADL assumes liability for any subsequent changes to be added as a property of the Statements to be able to perform an effective implementation of updates.

Validation of Statements in xAPI focuses exclusively on syntax, not on semantics. The rules apply to the structure and not the significance. Tutors are responsible for applying some rules to the valid meaning of definitions and activities.

In terms of structures for interactions or evaluations, as well as the extension of utility, they were borrowed from the SCORM model. These definitions are simple and offer a familiar utility. However, they can be extended by using extensions to that type.

xApi does not define how registration is accomplished, LRS is the provider or a delegated system that can ensure such a mechanism. Four sub-APIs compose xAPI:

- Statement - the basic communication mechanism;
- State - "this is a scratch area for Activity Providers that do not have their internal storage, or need to persist state across devices" [24];
- Agent Profile - this is an area where arbitrary key/documents are saved that are related to an Agent;
- Activity Profile - this is an area where arbitrary key/documents are saved that are related to an Activity.

Another goal pursued by xAPI is to allow tracking across multiple domains. For browsers that do not allow this property, there is alternate syntax via a specified request.

2.5. Statistical concepts used in the assessment of the proposed e-learning approach

In this work we will use a set of statistical concepts to evaluate our approach.

T-tests (hypothesis tests) are basic tests for the analysis of continuous data. A t-test allows us to compare the average values of the two sets of notes and determine if they have major differences between them.

The test takes a sample from each of the two sets and establishes the statement of the problem by assuming a null hypothesis that the two means of evaluation are similar. Based on the applicable formulas, certain values are calculated and compared with the standard values, and the assumed null hypothesis is accepted or rejected accordingly.

Basically, the test takes a sample from each of the two sets and establishes the statement of the problem by assuming a null hypothesis. Based on the applicable formulas, certain standard values are calculated and compared and the assumed null hypothesis is accepted or rejected accordingly.

If the null hypothesis qualifies to be rejected, it indicates that it is not due to chance. The t test is just one of many tests used for this purpose. Statisticians must use additional tests other than the t test to examine several variables.

Calculating a t-test requires three key data values. These include the difference between the mean values from each data set, the standard deviation, and the number of data values. The standard deviation is the measure of a spread of data around the mean [54].

When testing a hypothesis, it must be decided what difference between averages is needed to reject the null hypothesis. This level of significance can have different values; the most commonly used are 0.05 and 0.01. A level of 0.05 means that the average of our sample is significantly different from the average hypothesis if the chances of observing the average of the sample are less than 5% [55]. Similarly, it applies to any other level chosen.

In testing the significance of the null hypothesis, the p-value is also useful. This represents the probability of obtaining the test results at least as well as the results actually observed, assuming that the null hypothesis is correct [56]. A small p value means a great statistical significance of the observed situation.

The accuracy is the measure that show us how close we are to the true value. The accuracy has the formula:

$$\frac{TN + TP}{TN + FN + TP + FP}$$

In the formula above, we have the following:

- TN – true negative – a correct result in which the condition does not holds;
- FN – false negative – is a result, which wrongly indicates that a condition does not holds;
- TP – true positive – a correct result in which the condition holds;

- FP – false positive – an error in classification in which a result incorrectly indicates the presence of a condition.

The F score is a measure of the accuracy of a test. It is calculated from the accuracy and recall of the test, where accuracy is the number of correctly identified positive results divided by the number of all positive results, including those that were not correctly identified, and the recall is the number of correctly identified positive results divided by the number of all samples that should be identified as positive.

For an F test, the steps we need to follow are:

1. Affirmation of the null hypothesis and the alternative hypothesis.
2. Calculation of the value F. The value of F is calculated using the formula

$$F = (SSE1 - SSE2 / m) / SSE2 / n-k,$$

where SSE = residual sum of squares, m = number of constraints and k = number of independent variables.

3. Finding the F statistic (critical value for this test). The statistical formula F is:

F Statistics = group variation means / average of group variations.

4. Supporting or rejecting the null hypothesis.

The F score is often used in machine learning [5]. F-measure is the harmonic mean of recall and precision. The precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of the total amount of relevant instances that were retrieved.

The questionnaire is another research tool. It is composed of several questions related to the opinions, preferences, interests of the subjects in precise circumstances [57]. This is a method of collecting data through the questions asked. The questionnaire can include both closed and open questions. Closed questions involve a mandatory choice of one or more of the answer options, and open-ended questions accept any answer. It should be noted, however, that any closed question involves an answer type such as *I don't know*, *I don't answer*, in order to keep the respondent's right not to answer.

2.6. Comparison of Learning Objects

In this section, we summarize the main features of all presented learning objects and we compare them to motivate our research. We chose to compare the LOs models from several points of view.

The **accessibility** of LOs from our point of view refers to the possibility of ensuring their use by anyone, from anywhere, on different types of devices, facilitating the personalized teaching-learning activity.

Regarding the **content** of the learning object, it can be static or static combined with dynamic elements. By static content, we mean a text, an image, a video that is the same whenever the object is accessed. By dynamic content, we mean variables or symbols that are instantiated based on randomly generated values, so that each instance is a different exercise.

LO	Accessibility	Static content	Static content combined with dynamic elements	Automatic feedback containing the correct result	Automatic grading	Free from programming skills
IEEE Learning Objects	Yes	Yes	No	No	No	Yes
SCORM	Yes	Yes	No	Yes	Yes	No
CISCO	Yes	Yes	No	No	No	No
Lernativity	Yes	Yes	No	No	No	Yes
Netg	Yes	Yes	No	No	Yes	Yes
H5P	Yes	Yes	No	No	Yes	Just web programming
GLOs by T. Boyle	Yes	Yes	No	No	Yes	Yes
GLOs by Damasevicius and Stuiikys	Yes	Yes	No	No	Yes	Yes
Moodle coordinate question	Yes, in Moodle Platform	Yes	only numerical variables	Yes	Yes	Yes
Moodle calculated question	Yes, in Moodle Platform	Yes	only numerical variables	Yes	Yes	Yes
AGLO	Yes	Yes	Yes	Yes	Yes	No

Table 2.1. LO comparison

In terms of accessibility, the LO models shown in Table 2.1 are compatible with different types of devices such as phone, tablet, laptop, or PC.

IEEE learning objects are represented by any content used for learning or training. The content can be any text, images, audio. They have no feedback, no evaluation and require no programming knowledge.

SCORM is a set of technical standards that ensures that eLearning content works properly on an LMS platform. LO content is static, an electronic representation of media, text, images, audio, web pages or other data that can be presented in a web client. LOs can have automatic feedback and evaluation. SCORM allows tutors to distribute their content in a variety of LMSs, but to create this content it is necessary for the tutor to have a minimum of programming knowledge.

CISCO is a content model that restricts the number of learning components to seven. The LO content is also static, it can contain sentences or paragraphs, images, animations, etc. Cisco has a strategy for developing and implementing RLOs, but the tutor needs to have some basic programming knowledge.

Lernativity is a content model in which raw media elements, such as a single sentence or paragraph, illustration, animation, and others are grouped up to the course level. The LOs created with Lernativity contains only static content. How to combine objects at different levels of granularity follows a few rules, but you do not need programming skills to meet them.

The term LO used by NETg comprises three parts: a learning objective, an activity to teach the objective, and an evaluation unit that measures the objective.

These are abstract types, which can be mapped and aggregated on four levels, up to the course level. The content can be any text, images, audio. They have automatic evaluation. LOs do not require programming knowledge to be realized.

H5P is an open source tool focused on creating interactive HTML5 content. Depending on the type of task, the difficulty of creating it differs, so the process of creating interactive learning tasks requires minimal programming knowledge.

GLOs coined by Tom Boyle are reusable patterns. These patterns are accessible from any devices. The GLO-Maker are a tool that offers the tutors the possibility to create concrete LOs based on these patterns, by adding static content. The student's answer is appreciated with a grade.

The GLOs proposed by Vytautas Štuikys and Robertas Damaševičius are a model expanded with meta-programming technologies, quite general and accessible. The templates were regarding only to IT disciplines. The content is static, and the student response gets a mark.

Moodle coordinate questions and Moodle calculated questions are two plugins, in which the question is made from static content combined with dynamic numerical variables. The tutor can set an automatic feedback to each question. After evaluating the student's answer, a grade is returned. No programming knowledge is required to create such questions. The Moodle platform provides the necessary documentation for their realization.

By using AGLO, learning objects accessible from any device are obtained. The content is made of static text combined with variables, these variables make the statement dynamic. Subject-specific feedback can be made for each AGLO. The student's answer is automatically compared to the correct answer, the student had shown this comparison in the feedback section. Also, his answer is automatically given a grade. In order for a tutor to develop AGLO, he needs programming knowledge.

	Question	Answers	Variables types	Random values
Moodle coordinate question	Embedded variables and text	Is defined as mathematical expression	numbers, strings enclosed by quotes, list of numbers or strings	Are generated in an interval given by the tutor
Moodle calculated question	Embedded variables and text	Is defined as mathematical expression	numbers expressed in several formats	Are generated in an interval given by the tutor
AGLO	Embedded variables and text	Simple answers, multiple line answers, structures	Any basic type, and dynamic types created by composing basic ones facility supported by JavaScript format and its functions	Complex data structures generated from random values

Table 2. 2. Moodle plugins and AGLO comparison

Since the closest to the AGLO model are Moodle coordinate questions and Moodle calculated questions, we have chosen to make a more detailed comparison of them in order to motivate our research.

Regarding the question sections for both Moodle plugins and AGLOs, this is done by combining between static text and variables that are replaced with values during the instantiations.

The answer section for the Moodle plugins are calculated only on the basis of mathematical expressions. The answer in AGLO is domain specific library generated. The answer section for AGLOs includes several types of answers depending on the subject: numbers calculated based on formulas or functions, strings, different types of objects represented as strings such as linked linear lists, double-linked lists, and intervals.

In the Moodle coordinate questions, the variables are:

- numbers expressed in several formats;
- strings enclosed by quotes;
- lists of numbers associated to arrays;
- lists of strings associated to arrays;
- algebraic variables - as a set of numbers, defined in the non-random variable

scope.

In the Moodle calculated questions, the variables are only numerical integer or float.

In AGLOs, the variables are:

- numbers expressed in several formats;
- strings enclosed by quotes;
- lists of numbers associated to arrays;
- lists of characters associated to arrays;
- lists of strings associated to arrays;
- objects: intervals, fractions, list nodes, linear linked lists, double linked lists;
- trees;
- graphs;
- SVG representations.

From this point of view, we can say that AGLO allows a greater variety of types of variables, thus allowing the approach of a wider area of notions.

2.7. Summary

In this chapter, we presented different definitions of LOs, and existing standards. Since there are several accepted definitions for learning objects, each standard has been developed based on a certain direction. The IEEE Standard focused more on the metadata characteristics of a learning object. SCORM is a standard that guarantees compatibility with almost any LMS. In the case of the CISCO model, the emphasis is on learning objectives, each LO having an objective whose achievement can measure it. Dublin Core Standard has focused on networking educational resources. The Learnativity Content model and NETg model focused on LO granularity

and rules on their combination depending on the level of granularity. ALOCoM was built as a generalization of existing Learnativity, SCORM, and NETg standards. H5P is a newish tool created to facilitate the development of attractive HTML5 content.

In order for these LOs to be distributed, LMSs are needed. We presented three of the most used platforms in education, namely Moodle, Ilias, and Blackboard Learn.

Next, we presented a new generation of LOs, namely GLOs. GLO are templates that can be reused and filled with content. Regarding the GLOs, there are two clear directions, one by Tom Boyle and one by Vytautas Štuikys and Robertas Damaševičius. This category of learning objects also includes the plugins offered by Moodle, namely Moodle Coordinate Questions and Moodle Calculated Questions.

Because the maximum potential of LOs has not been reached, there have been several projects to develop this field. Thus further, we the LOs grouped in sequences, the LOs improved by object-oriented design and CLAVIRE e-Science platform.

We also present several statistical concepts that we use in chapter seven to evaluate our approach. Finally, we made a comparison of LOs.

We concluded that AGLO is a model worth developing because it offers advantages in terms of the notions it can target. It is also a plus that it can provide automatic feedback and automatic grading of the student's response.

3. THE STRATEGY FOR MEETING THE OBJECTIVES

The field of e-learning and LOs is a wide one. There are several accepted models, each of them aiming at one specific direction, namely pedagogical, technical, or economic. Following the study on LOs, we identified that they are difficult to reuse and difficult to adapt to each discipline. Also, the content of the LOs is static, the student has the same exercise at his disposal each time he accesses the LO. Consequently, we will study the possibility of using the AGLO model for different STEM disciplines. We will especially consider making templates based on random numbers that give the student the opportunity to have a different exercise at each instance, automatic response assessment and feedback.

Regarding the realization of this thesis, we divided the activity carried out into five main tasks:

1. Realization of an abstraction of the AGLO templates;
2. Model design for AGLOs;
3. Development and implementation of the model for STEM disciplines;
4. Development of domain-specific JavaScript libraries;
5. Evaluation of the model by applying the AGLOs in the learning-evaluation process in the classroom.

Regarding the realization of an abstraction of the AGLO templates, we consider that it is necessary to create an algorithm. It must include well-defined and structured steps. The algorithm must be easy to understand. Its application must be able to be done by any tutor who wants to make AGLO templates. This task will help us to fulfil the second thesis objective, namely to propose a step-by-step methodology based on abstractions to create reusable AGLO templates.

We will use the already existing AGLO model. The templates will be written in a XML form, and the root element will be the action element. Each template will be organized into six sections, namely name, scenario, theory, question, answer, and feedback.

The development and implementation of the model involve the creation of AGLO templates for different notions. The targeted notions will be chosen from several STEM disciplines, namely Mathematics, Data Structures and Algorithms, Algorithm Analysis and Design, and Operating Systems. For the Mathematics, notions we will focus our attention on fractions, operations with intervals, equations, and inequations involving absolute values, and using abbreviated calculation formulas. For the Data Structures and Algorithms discipline, we will focus our attention on searching algorithms, namely linear search, linear search with sentinel, binary search and search by interpolation, sorting algorithms, namely insertion sort, selection sort, bubble sort, shell sort, and quick sort, and linked lists, namely linear linked lists and double linked lists. For the Algorithm Analysis and Design discipline, we will focus our attention on some basic notions about trees and graphs. For the Operating Systems discipline, we will focus our attention on basic Linux commands. We will analyze the chosen notions. We will identify the learning objectives. We will create examples that target the objectives. We will abstract the examples in an AGLO template. Through this task, we

will be able to reach the fulfillment of the third objective of the thesis, namely to abstract multiple learning concepts from STEM disciplines.

In the development of domain-specific JavaScript libraries, it is necessary to model the targeted notions. These libraries will contain the builders of abstract objects that will be identified with the notions, as well as functions and methods for working with them. By reaching these tasks, our scientific research will reach a TRL3 level [58]. This task will help us to fulfil the fourth thesis objective, namely to develop domain-specific libraries modeling concepts to assist the instantiation of AGLOs.

Applying AGLOs in the classroom learning assessment process will allow us to evaluate the model. By applying specific statistical metrics, we will be able to measure the accuracy of the model. Applying our model in the classroom, we will be able to consider it will reach a TRL4 level. Through this task, we will be able to reach the fulfillment of the fifth objective of the thesis, namely to validate our AGLO approach in practice, on groups of students showing their effectiveness.

The thesis objectives	The thesis chapters
Objective O1 is to make a bibliographic study of LOs and a comparative analysis of the studied models	Chapter 2
Objective O2 is to propose a step-by-step methodology based on abstractions to create reusable AGLO templates.	Chapter 4
Objective O3 is to abstract multiple learning concepts from STEM disciplines	Chapters 5 and 6
Objective O4 is to develop domain-specific libraries modeling concepts to assist the instantiation of AGLOs	Chapter 7
Objective O5 is to validate our AGLO approach in practice, on groups of students showing their effectiveness	Chapter 8

Table 3. 1. The thesis objective associated with the thesis chapters

In table 3.1, we present the associations between the thesis objectives and the thesis chapters. Each chapter targets a proposed objective.

Accomplishing the proposed tasks will lead us to obtain the thesis objectives.

4. THE AUTO-GENERATIVE LEARNING OBJECT MODEL

The chapter presents the abstraction process that we applied. The steps of the abstraction algorithm and their application in creating scenarios for AGLO for STEM disciplines are presented.

The structure of the model and the mathematical definition are presented. Semantics is also explained using a concrete example.

4.1. AGLO creation by StepWay Abstractization and generalization

In this chapter we will present the abstraction algorithm for obtaining AGLOs and we will present the application of this algorithm in different situations.

4.1.1. The abstraction algorithm

The process of making these AGLOs includes several stages:

Step 1: The learning objective identification.

Step 2: The concrete exercise identification.

Step 3: The variables identification.

Step 4: Variables type and range identification.

Step 5: Primary input data identification.

Step 6: Building a computational scenario.

Step 7: The domain specific library objects and methods identification.

Step 8: Setting up the difficulty level.

Step 9: Necessary intermediate variables.

Step 10: Answer variables and their computation formula identification.

Step 11: Testing the object.

In the first step, the tutor extracts specific competencies and learning objectives that the use of AGLOs can accomplish.

In the second step, the tutor creates an example of an exercise or problem, such as the ones he would teach in class, which would correspond to the previously identified goal.

Then follows the generalization of the chosen exercise. In performing this step, consideration is given to determining all the variables needed to create an exercise that meets the established learning objective.

For each variable its type is established, these types can be integer, float, character, string, or objects of fraction type, interval, tree, SVG. Where appropriate, the interval in which it can take variable values is also written, this being of the form [minimum, maximum].

In the next step, the input data are identified. These ones will be randomly generated at the chosen intervals. Functions such as `random_int()`, `random_float()`, `random_array()`, `random_linked_list()` are used for random generation. The generalization also includes determining how we can change certain variables types in the exercise.

After that, we make sketches about making templates. These templates must be designed so that they can be implemented as existing exercises meet the desired learning objective. When constructing a calculation scenario we also consider the mode in which the dependencies between the variables are created.

In the next stage, we identify the required functions and methods necessary and we decide what domain-specific JavaScript libraries it is needed.

In the next step, we set up the difficulty level by modifying the variables interval.

Next, we will identify how to calculate intermediate variables. Not all exercises will have such variables.

In the next step, we will determine how we will calculate the answer. There are cases in which it will be calculated based on formulas, sometimes based on methods, and sometimes directly in the scenario.

The next step includes the implementation of the template in an XML file and testing the obtained object. In this stage, the tutor tests the template made by several snapshots. It follows the degree of difficulty of calculating the answer and the inclusion between variants of special cases, if they exist. Each template is refined, as far as the design allows.

4.1.2. The abstraction process applied on an example

To exemplify the way in which abstraction process is applied, we have chosen as a model an AGLO regarding the double-linked linear lists. Next, we will present the abstraction process step by steps.

Step 1: We chose the following as a learning objective: the student can recognize the successor or predecessor of a node in a double-linked linear list.

Step 2: To accomplish this goal in class we would present to the student a double-linked linear list from which we will choose a node and ask him to identify his predecessor or successor, see Figure 4.1.

Step 3: In this stage we have identified the necessary variables, namely in an integer that will represent the number of nodes in the list, a list with n nodes, an integer and that will represent the order of a node in the list, an option and the answer desired.

Step 4: in this situation, we have the following variables:

- n , the number of nodes in the linear list;
- an list object;
- a graphical representation of the list;
- a number, i , what will identify the chosen node;
- a string representation of the chosen node;
- an option, allowing the choice of a successor or predecessor;
- a node object that represents the desired answer;
- a string representation of the answer.

Regarding the intervals of the variables, we have n , an integer, between 3 and 10, and i , an integer, between 1 and $n-2$ and an option, which can have the values 0 or 1.

Step 5: We identify as input data the number of nodes, the double-linked linear list, the identification number of the chosen node and the option. Thus, the number of nodes, the identification number of the chosen node and the option will be generated using the `random_integer()` function, and the list is an object whose values will be generated randomly. We will choose the node information to be of type integer or char.

Step 6: We made a sketch about this template, see Figure 4.1. In this scenario, we realize that both the creation of the list and the choice of the node depend on the number of the integer n . In order for everything to be as clear as possible, the graphical representation of the list is necessary.

Step 7: Given what we identified in the previous step, we will need the constructor of a linear list object and specific methods to represent graphic a linear list object.

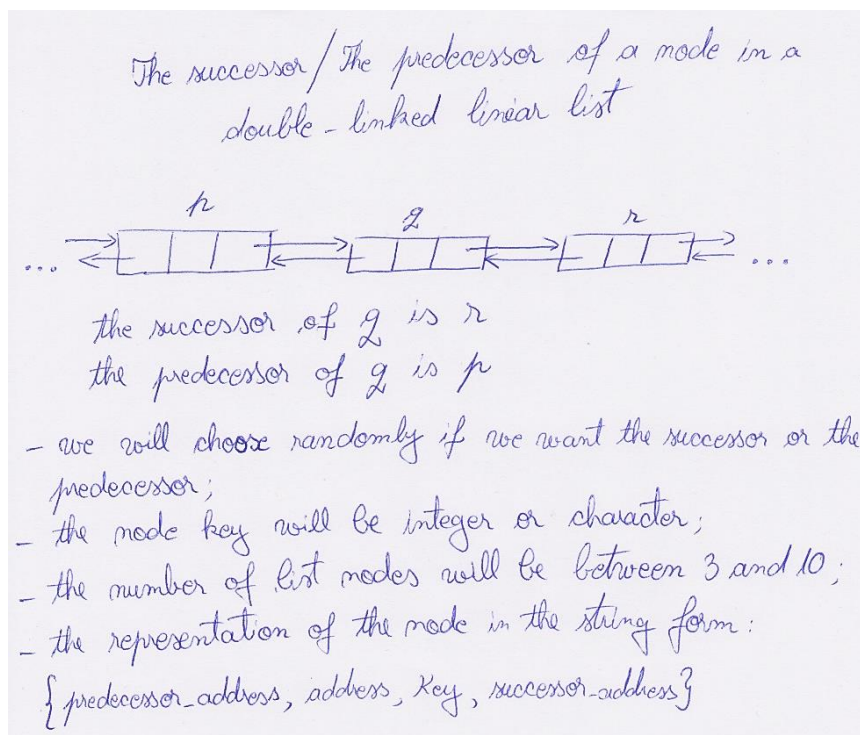


Fig.4.1. Sketch for making an AGLO

Step 8: Regarding the level of difficulty of the exercise, we consider the chosen intervals appropriate.

Step 9: We do not consider that in this case we need other intermediate variables.

Step 10: Regarding the answer, we have first the node type object and then its representation in the form of a string.

Step 11: After testing the AGLO, we realized that the number of nodes should not exceed seven due to the space occupied by the graphical representation. Thus we modified the interval in which the number of nodes is generated from [3; 10] to [3; 7].

4.1.3. Abstractions for different concepts from STEM disciplines

Next, we will present from each discipline the way in which we abstracted an AGLO.

For Arithmetic, regarding the work with fractions we will show the steps through which we realized the AGLOs that aim at the addition of fractions.

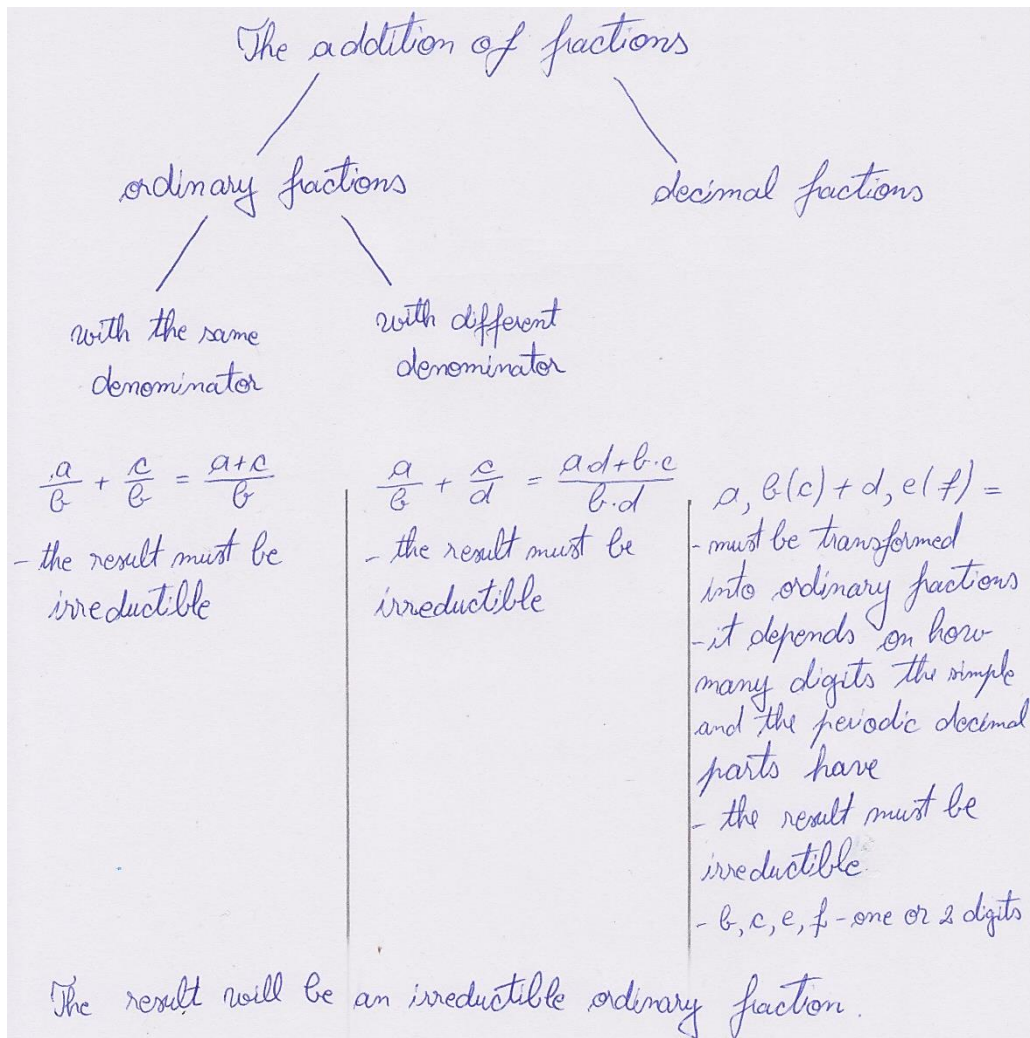


Fig.4.2. Sketch for making the fractions addition AGLOs

Step 1: The learning objective is the student to know how to add two fractions.

Step 2: Regarding the addition of fractions, we identified three types of possible exercises, namely: the addition of ordinary fractions with the same denominator, the addition of ordinary fractions with different denominators and the addition of decimal fractions. Thus, we will make three different types of templates see Figure 4.2.

For the addition of the ordinary fractions that have the same denominator, we further performed the following:

Step 3: The identified variables are two fractions for which we need two counters, a common denominator, and the fraction that represents the sum.

Step 4: For numerators we chose the interval $[1; 50]$, and for the denominator the interval $[20; 90]$.

Step 5: As input data, we identified the two counters and the common denominator.

Step 6: Regarding the addition of fractions that have the same denominator, add the two counters, and the denominator is the common one.

Step 7: In this case, we need a single fraction object and the method of determining the common divisor to turn the fraction into an irreducible one.

Step 8: Regarding the level of difficulty, we considered that the intervals chosen in step 4 are good.

Step 9: The required intermediate variable is the fraction object that represents the sum of the two initial fractions.

Step 10: To calculate the answer variable we chose to simplify the sum fraction with the common divisor of its denominator and numerator.

Step 11: Testing the object showed us that the chosen scenario meets our goal.

For the addition of the ordinary fractions that have different denominators, we further performed the following:

Step 3: The identified variables are two fractions for which we need two counters, and two denominators, and the fraction that represents the sum.

Step 4: For numerators we chose the interval $[1; 20]$, and for the denominator the interval $[2; 20]$.

Step 5: As input data, we identified the two counters and the two denominators.

Step 6: Regarding the addition of fractions that have different denominators, we considered it necessary to generate two fraction type objects and to apply the sum method.

Step 7: In this case, we need three fractions object, the method to add and the method of determining the common divisor to turn the fraction into an irreducible one.

Step 8: Regarding the level of difficulty, we considered that the intervals chosen in step 4 are good.

Step 9: The required intermediate variable is the fraction objects.

Step 10: To calculate the answer variable we chose to simplify the sum fraction with the common divisor of its denominator and numerator.

Step 11: Testing the object showed us that the chosen scenario meets our goal.

For the addition of the ordinary fractions that have different denominators, we further performed the following:

Step 3: The identified variables are two fractions for each of them we need the integer part, the simple decimal part and the periodic decimal part, as well as the fraction that represents the sum.

Step 4: We have chosen for the variables to be randomly generated the following restrictions:

- the whole part will be an integer with one digit;
- the non-periodic decimal part and the periodic decimal part will be generated in the interval [1; 15].

Step 5: In this example the input data are the three randomly generated numbers that form the two decimal fractions.

Step 6: Regarding the addition of decimal fractions we generated two fraction type objects and applied the sum method.

Step 7: In this case, we need three fraction type objects, the method to add and the method to determine the common divisor to turn the fraction into an irreducible one.

Step 8: Regarding the level of difficulty, we considered that it is enough that the whole part and the periodic decimal part have a single digit.

Step 9: The required intermediate variables are the objects of the fraction.

Step 10: To calculate the answer variable we need to simplify the sum fraction with the common divisor of its denominator and numerator.

Step 11: Following the implementations performed, we considered the changes:

- the non-periodic decimal part will be generated in the interval [6; 14]
- the periodic decimal part will have a single digit between 1 and 5.

We made these changes because it happens in some generations that the periodic decimal part is equal to the non-periodic decimal part, which was not mathematically correct. In addition, if the periodic decimal part has two digits, large numbers are obtained, and we want the emphasis to fall on practicing and not on having a very high difficulty of the exercise.

Regarding the mathematics for the eighth grade, we chose to present the way in which we realized the AGLO for solving the equations involving absolute value.

Step 1: The chosen learning objective is for the student to know how to solve linear inequations involving absolute values.

Step 2: For this the student must know how to solve an inequation of form $|ax + b| \leq c$.

Step 3: In this case we need the terms a, b, c, and the interval that represents the solution.

Step 4: For the variable a, since it will have the role of divisor, we will choose only values that allow obtaining finite decimal numbers, namely 2, 4, 5, 8, 10, 20. The variable b will have values in the range [-50,+50]. The variable c can have only positive values, so it will be in the range [1,200]. The variable that will represent the solution will be of type randomInterval object.

Step 5: The variables a, b and c represent the input data, and they will be generated randomly in the chosen intervals.

Step 6: Based on the generated variables, the solution of the equation will be calculated.

Step 7: For the response variable we will use the object we have defined in Interval.js domain specific library.

Step 8: Regarding the level of difficulty, the restriction applied for the variable a is sufficient.

Step 9: As necessary intermediate variables we will need the set consisting of the values chosen for the variable a and an index that will specify the value chosen from the set.

Step 10: The variable answer will be calculated according to the formula, the left bound will be $\frac{-c-b}{a}$, and the right bound will be $\frac{c-b}{a}$.

Step 11: After we implemented the object and made some snapshots we considered that it corresponds to the intended objective.

Regarding Data Structures and Algorithms disciplines, we chose to present the way we applied the abstraction proces in the realisation of the template for the steps of the linear search algorithm.

Step 1: The intended learning objective is for the student to know the steps of the linear search algorithm.

Step 2: In this case in class the student would have received an array and a value to look for it, item by item. The searched value may or may not exist in the table.

Step 3: To make this scenario we need a table, the number of elements in the table, a search value and a variable to keep the steps of the algorithm.

Step 4: The number of elements in the table, the elements of the table and the searched value will be integers. The steps of the algorithm could be stored in a string variable. The number of elements will be chosen in the range [3; 10], the values of the elements in the table and the element to be searched in the range [1; 300].

Step 5: The number of elements in the table, the elements of the table and the searched value will represent the input data for this AGLO.

Step 6: To remember all the steps of the algorithm in a string, we chose to make a method for the linear_search object.

Step 7: So we need a linear_serch object and the getSteps () method.

Step 8: Regarding the difficulty of the exercise, I chose that the value to be searched should be part of the elements of the table.

Step 9: Because we have chosen that the value to be searched is part of the elements of the table, it will not be generated from a range but will be chosen randomly from the elements of the table. Thus we will need a variable to indicate its position in the table. The linear_search object variable can also be considered as an auxiliary variable.

Step 10: The answer will be a string returned by the getSteps () method.

Step 11: After testing the object, we modified the interval in which the number of elements in the table is chosen to [6; 11]. We made this change because in the initial version few steps were generated and thus the algorithm may not be well understood.

For Algorithm Analysis and Designed Discipline, we will show the steps through which we realized the AGLO that aim at the recognition of the nodes on each level of a tree.

Step 1: The intended learning objective is for the student to know how to recognize the nodes on each level of a tree.

Step 2: Given a randomly generated tree, the nodes on each level are identified and write in order.

Step 3: We need a randomly generated tree, the number of nodes and the maximum degree of a node and the response variable.

Step 4: The number of nodes will be an integer between 6 and 10. The maximum degree of a node will be an integer in the range [2; 4].

Step 5: The number of nodes and the maximum degree of a node are required for random generation of a tree.

Step 6: We considered that an easy way to represent the fulfillment of the requirement is to write the nodes on each level on a line.

Step 7: To create this scenario you need a tree object and a method to return the desired answer.

Step 8: Regarding the level of difficulty, we considered that the restrictions from step 4 are sufficient.

Step 9: The required intermediate variable is the representation of the tree in the SVG form. We consider that the representation of the tree is very important in understanding the notion.

Step 10: The response will be generated by the implemented method `tree_levels()`.

Step 11: After testing the object, we consider that it meets the intended learning objective.

Regarding the Operating Systems discipline, we chose to present the way in which we realized the AGLOs for the commands to create a hierarchy of directories.

Step 1: The learning objective we pursue is for the student to create a hierarchy of principals. In Linux this can be done in two ways with or without the `-p` option. In this situation, it is necessary to create two templates, one for creating a folder hierarchy using the `-p` option and one for creating a folder hierarchy without the `-p` option. Thus, below we will present two scenarios.

Step 2: In the first case, a folder hierarchy is given in the form of a tree:

```
alpha
* beta
**gamma
***delta
***omega
```

This is created using the `-p` option as follows:

```
mkdir -p alpha / beta / gamma / delta /
mkdir -p alpha / beta / gamma / omega /
```

Step 3: In this case we need a hierarchy of folders. Because it is generated like a tree, it needs the number of nodes, ie the number of folders in the hierarchy, and the maximum degree of each node, ie the maximum number of subfolders that a folder can have.

Step 4: The number of folders will be randomly generated in the range [5.8], and the maximum number of subfolders will be 2 or 3.

Step 5: The input data will be the two for which we set the generation intervals in the previous step.

Step 6: In this case, based on the input data, a folder tree will be randomly generated.

Step 7: We will need an object of type FolderTree and a method to return the response as a string toMkdirPathsString ().

Step 8: We consider that the level of difficulty obtained by the restrictions imposed on the previous steps is sufficient.

Step 9: As an intermediate variable we need the representation of the folder hierarchy as a string.

Step 10: The answer will be returned by the method performed.

Step 11: After testing the object, we did not make any changes, the objective being fulfilled.

Step 2: In the second case a hierarchy of folders in the form of a tree is given:

```
alpha
* beta
**delta
*gamma
**omega
```

This is created without the -p option as follows:

```
mkdir alpha
cd alpha
mkdir beta
cd beta
mkdir delta
cd..
mkdir gamma
cd gamma
mkdir omega
cd..
cd..
```

Step 3: And in this case we need a hierarchy of folders, a number of folders and the maximum number of subfolders that a folder can have.

Step 4: The number of folders will be randomly generated in the range [5.8], and the maximum number of subfolders will be 2 or 3.

Step 5: The input data will be the two variables presented in the previous step.

Step 6: Based on the input data, a folder tree will be randomly generated.

Step 7: We will need an object of type FolderTree and a method to return the response as a string toMkdirCdPathsString ().

Step 8: We consider that the level of difficulty obtained by the restrictions imposed on the previous steps is sufficient.

Step 9: As an intermediate variable we need the representation of the folder hierarchy as a string.

Step 10: The answer will be returned by the specific method performed.

Step 11: After testing the object, we consider that it meets the intended learning objective.

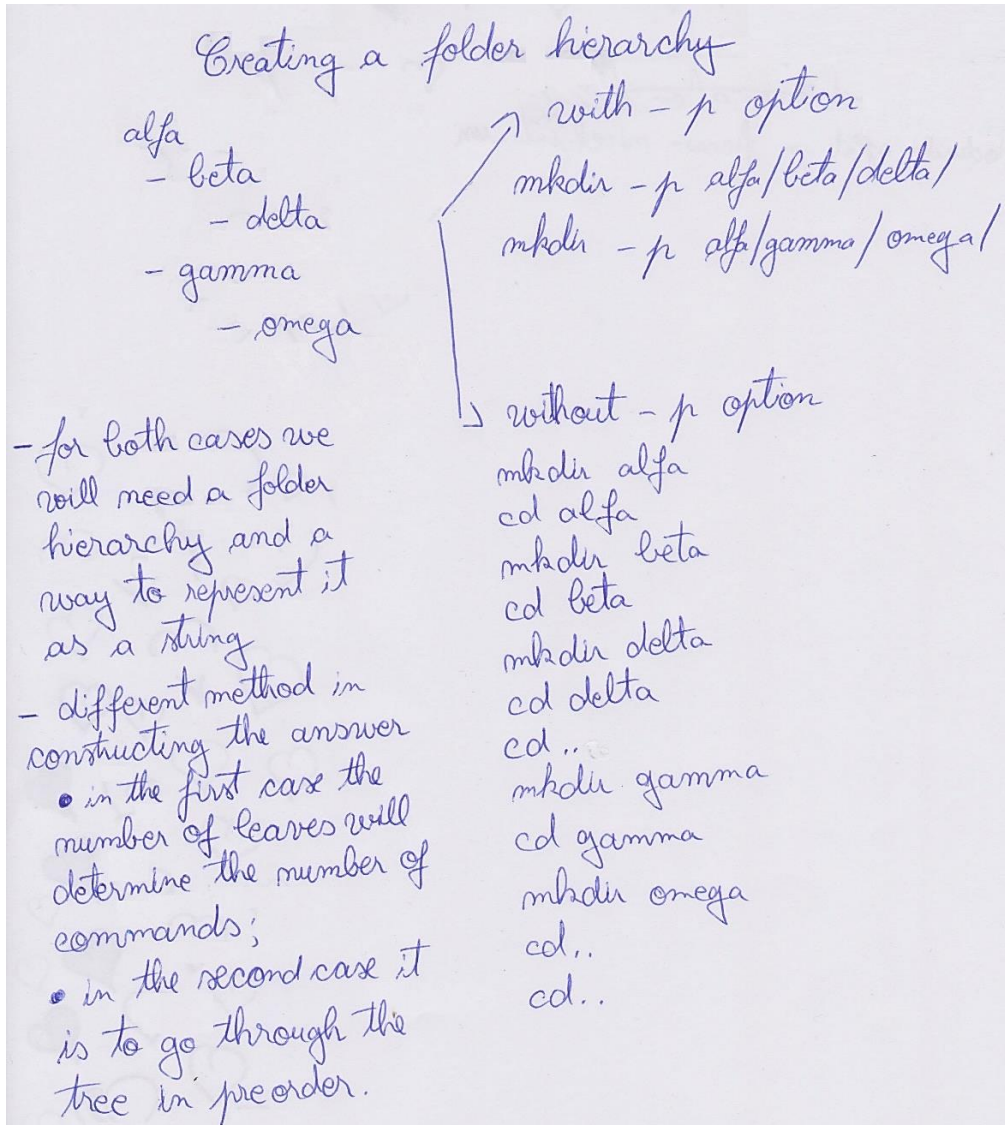


Fig.4.3. Sketch for making the creation of a hierarchy of directories AGLOs

4.2. Mathematical Definition and Structure

Next, we will define the mathematical model of AGLO instantiated with random numbers. All functions and operators from the JavaScript language standard [59] are indicated by F_{JS} .

Any function $f_i \in F_{f_s}$ has an argument of form $x = \langle x_1, x_2, \dots, x_n \rangle$. The value of n represents the number of arguments for the x vector.

The f_i functions can have as argument values:

- integers located within a certain range;
- doubles (floating-point values) also located within a certain range;
- strings as vectors of character;
- arrays;
- objects.

The expression construction function e_i is defined as:

$$e_i(x) = (f_1 \circ f_2 \circ \dots \circ f_k)(x),$$

where x can be a constant value or a randomly generated value.

We define AGLO as a quintuple:

$$Ag = \langle S, T, Q, A, F \rangle,$$

where each component is a section [60].

The scenario section S is a vector of composed functions:

$$S = \langle e_1(x), \dots, e_{n_s}(x) \rangle.$$

The theory section T is made from constant values

$$T = \langle t_1, t_2, \dots, t_{n_t} \rangle, t_i \in [0, 255] \cap \mathbb{Z}.$$

The question sections Q , the answer section A , and the feedback section F are vectors of random generated value and constant values.

Structurally AGLO model has six sections: name, scenario, theory, question, answer, feedback [60].

The name section is the first XML element and it is reserved for the name of AGLO. This section is made up of a single sentence that describes the competency (framework objective) covered by that exercise.

The scenario section is the section where the variables are defined and initialized. This section is divided into two. The first part consists of a text - static text - in which the mode of operation of the LO is presented. The second part consists of declaring and initializing the symbols necessary to solve the problem presented in the static text. Symbols can be initialized based on random numbers or based on ones already initialized through some specific functions.

The theory section is made by static text - a text that does not change at each instance. In this section, the student can read a brief presentation of the theory needed to solve the next problem.

The question section is the section where a task is placed. The statement is made of static text blended with randomly instantiated variables.

The answer section is where the input is read from the student and is assessed automatically.

The feedback section is composed out of static text, which has the value to explain the student the essence of the exercise if it is necessary. For some LO feedback, it also contains some variables needed to improve the learning process. Moreover, at the end the student's answer is compared on components with the correct computed answer.

```

01 AGLODef ::= "<action>" Name Scenario [Theory] Question
              Answers Feedbacks "</action>"
02 Name ::= "<name>" (ID)* "</name>"
03 Scenario ::= "<scenario>" [ Comment ] Symbol* "</scenario>"
04 Comment ::= (ID|CT)*
05 Symbol ::= "<symbol>" SymbolName Type Expression "</symbol>"
06 SymbolName ::= "<name>" ID "</name>"
07 Type ::= "<type>" ("boolean" | "int" | "float" | "double" |
              "string" | "array") "</type>"
08 Expression ::= "<expr>" Function "(" ExpressionList ")" "</expr>"
09 ExpressionList ::= Expression (, Expression)*
10 Function ::= (element from functions and operators list of JavaScript)
11 Theory ::= "<theory>" (ID)* "</theory>"
12 Question ::= "<question>" (ID | Value)* "</question>"
13 Value ::= "<value>" "<name>" ID "</name>" "</value>"
14 Answers ::= "<answers>" (Answer)+ "</answers>"
15 Answer ::= "<answer>" "<id>" INTEGER_LITERAL "</id>"
              (ID | Value)* Correctness "</answer>"
16 Index ::= INTEGER_LITERAL
17 Correctness ::= "<correct>" ("true" | "false") "</correct>"
18 Feedbacks ::= "<feedbacks>" (Feedback)+ "</feedbacks>"
19 Feedback ::= "<feedback>" AnswerIdList (ID | Value)*
              Active "</feedback>"
20 AnswerIdList ::= "<AnswerIdList>" (INTEGER_LITERAL)+ "</AnswerIdList>"
21 Active ::= "<active>" ("true" | "false") "</active>"

```

Fig.4.4 The Grammar

The model is defined using the EBNF meta-language [60]. The grammar thus obtained is represented in Figure 4.4.

In all the sections by ID, we denote the identifier token.

In the scenario section by CT, we denote any constant or literal: integer, float, character, or string. In the scenario, are defined symbols that act as instantiation parameters for the AGLO to be referenced in the next sections. Such a symbol has several properties as a name, a type, and an initialization expression based on random numbers.

The expressions are made up of functions and several operators unary or binary. To simplify implementation and increase portability, we rely on JavaScript expressions [60] that can be evaluated with the JavaScript function "eval". These expressions are the randomly controlled instantaneous key of the number-based object. Expressions can call native JavaScript functions, as well as user-defined functions, gathered in domain-specific libraries.

The theory section contains only static text about the learned concept in HTML format.

The question, answers, and feedbacks sections contain a mix of static data and dynamic values referring to the previously computed symbol values. The semantic of these sections is to create a dynamic content to be presented to the learner. Formalisms are also present, like:

- correctness - because we need to know which is the correct answer;
- feedbacks answer identifier list - because feedbacks are related to answers and when there are multiple answers only the related feedback is showed to the learner.

4.3. Semantics

In this subchapter, we describe in detail an AGLO model example. The chosen example refers to the rise an ordinary or decimal fraction to a power.

The root XML element is the action element. The first XML element is reserved for the analyzed AGLO name that is displayed to the learner for location and selection purposes. In this case Raising a fraction to a power.

The second XML element is the scenario element that contains a text description of AGLO and a set of symbols. The description is expressed in natural language and contains the stages of the exercise. An ordinary fraction and a decimal fraction are randomly generated; a number that will represent the power and an option depending on which one will decide which fraction will rise to power. The power fraction is calculated and its counter and denominator are displayed.

Then several symbols are defined. The first two symbols, *nr* and, *num*, represent the denominator and the counter of the ordinary fraction. The next two symbols, *pi* and, *pz*, represent the whole and the decimal part of the simple periodic decimal fraction. The fifth symbol, *putere*, represents the exponent of power; the sixth symbol, *optiune*, represents the option depending on which one will decide which fraction will represent the basis of power. The first six symbols are generated randomly, and these will be used to generate the next symbols.

The seventh and eighth symbols, *f1* and *f2*, are the two generated fractions. Each symbol is initialized by calling the specific constructor.

The ninth and tenth symbols, *var1* and *var2*, represent the writing of these fractions and will be used later in the question. The eleventh symbol, *intrebare*, is what will appear in the question section. And the twelfth symbol, *raspuns*, is what the answer should contain and will also be used in the feedback section.

In the following, we will discuss the XML elements depicted in Figure 4.5. The theory element consists only of static text. It presents to the student the notion to be applied in this exercise. The power of a number is obtained by multiplying the number with itself by as many times equal to the exponent.

The following element presents the question to the student. The questions section will refer to the generated fractions and the power exponent. The reference is implemented by using qualified `<value>` XML elements with name attributes. The semantics of these elements is that each element will be replaced with their corresponding value, so the content of e-learning will be dynamic. Each running session will have a different fraction and a different power.

The `<answers>` XML element is composed of several individual `<answer>` XML elements. Each `<answer>` element will be a mixture of static and dynamic values. In the current AGLO example, we used only one answer which is the correct one and in which there is only one dynamic variable. The calculated values and the completed values are compared and a result is issued which is transmitted to the learner.

```

<action>
<name>
  <text>
Ridicarea la putere a unei fractii
  </text>
</name>

<scenario>
  <text>
Se genereaza aleator o fractie ordinara si una zecimala, un numar care va
reprezenta puterea si o optiune in functie de care se va decide care fractie
se va ridica la putere.Se calculeza fractia putere si se afiseaza numaratorul
si numitorul acesteia.
  </text>
  <symbol name="nr" type="integer"> random (1,10,0);</symbol>
  <symbol name="num" type="integer"> random (1,10,0);</symbol>
  <symbol name="pi" type="integer"> random (1,6,0);</symbol>
  <symbol name="pz" type="integer"> random (1,9,0);</symbol>
  <symbol name="putere" type="integer"> random (2,5,0)</symbol>
  <symbol name="optiune" type="integer"> random (0,1,0)</symbol>
  <symbol name="f1" type="fractie"> fractieAleatoare2 (v ("nr"), v
("num")); </symbol>
  <symbol name="f2" type="fractie"> fractieAleatoare3 (v ("pi"), "", v
("pz")); </symbol>
  <symbol name="var1" type="string"> v ("nr") + "/" + v ("num");
</symbol>
  <symbol name="var2" type="string"> v ("pi") + "." + v ("pz")
+");</symbol>
  <symbol name="intrebare" type="string"> v ("optiune")? v ("var1") :
v("var2"); </symbol>
  <symbol name="raspuns" type="string"> v ("optiune")?
v("f1").laPuterea(v("putere")) : v("f2").laPuterea(v("putere")); </symbol>
</scenario>

```

Fig.4.5. AGLO Name and Scenario

```

<theory>
  <text>
Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de
ori egal cu exponentul.
  </text>
</theory>

<question>
  <text>
Ridicati fractia <value name="intrebare"/> la puterea <value name="putere"/>
  </text>
</question>

```

Fig.4.6. AGLO Theory and Question

The feedback section consists of several dynamic content elements, but also static statements. In our AGLO example, we help the student by providing repetition of the notion that was to be applied and the correct answer that was to be given.

```

<answers>
  <answer>
    <text>
<value name="raspuns"/>
    </text>
  </answer>
</answers>

<feedbacks>
  <feedback>
    <text>
Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de
ori egal cu exponentul. <br/>
Trebuie sa obtineti fractia <value name="raspuns"/>
    </text>
    <active> true</active>
  </feedback>
</feedbacks>
</action>

```

Fig.4.7. AGLO Answer and Feedback

4.4. Summary

In order to achieve an AGLO that aims at a certain learning objective, it is necessary to follow a few steps. In this chapter, we have presented the abstraction algorithm to perform AGLOs that we created. Next, we presented how we applied this algorithm for examples from each discipline Middle School Arithmetic, Data Structures and Algorithms, and Operating Systems.

We further, presented the mathematical definition and structure of the AGLO model. We presented and described the six sections of the model, namely name, scenario, theory, question, answer, feedback. Next, we describe in detail the AGLO semantics applied to a concrete example from Middle School Arithmetic.

5. MODELS FOR MIDDLE SCHOOL ARITHMETIC

As for the Arithmetic, we chose to approach the subject of fractions [5], of intervals and of abbreviated calculation formulas. We chose these chapters because it includes an important part of elementary notions that can be well-mastered by students if they repeatedly apply the theory necessary in exercises. The exercises resulting from the instance of the learning objects proposed by us follow the specific competencies present in the current school curriculum.

5.1. Fractions AGLOs

Regarding the topic of fractions, we have implemented twenty-one exercises that target classification, comparison, amplification, and simplification, all operations, transformations from decimal fractions into ordinary irreducible fractions, as well as the introduction of the whole in a fraction and the removal of the wholes from fractions.

To achieve these goals we have developed the library module “fractions.js”. In this module, fractions are modeled as JS classes, constructors, as well as methods necessary for the operations with fractions.

An ordinary fraction is initialized by the constructor function `randomFraction2()`, and a periodical mixed fraction is initialized by the constructor function `randomFraction3()`. Both constructors return a fraction object.

5.1.1. Fractions Classification AGLO

The first AGLO approaches the fractions classification. From the pedagogical point of view, this exercise aims to familiarize the student with basic mathematical notions as a proper fraction and improper fraction. The name section contains a text which includes the idea of exercise “The fractions classification”.

The second section is captured in Figure 5.1. This section contains two parts. The first part consists of a text that describes in natural language what has to be done in this exercise.

- a fraction is generated randomly;
- the denominator is compared to the counter;
- the following variants are obtained:
 - if the counter is greater than the denominator, the fraction is proper;
 - if the counter and denominator are equal, then the fraction is equal to the unit;
 - if the counter is lower than the denominator, the fraction is improper.

The second part includes the declaration and the initialization of all variables that are needed to solve the exercise. As shown in the previous figure, the counter and the denominator are initialized with two randomly generated numbers between 1 and 20. In this way, a random fraction is created and kept in the f symbol.

The *raspuns* answer symbol retains the correct answer that is returned by the *classify()* function. This symbol value will be compared to the student's input.

```

<scenario>
  <text>
    <pre>
      Se genereaza aleator o fractie ordinara. Se compara numaratorul si
      numitorul obtinandu-se urmatoarele variante:
      - daca numaratorul este mai mare decat numitorul fractia este supraunitara;
      - daca numaratorul si numitorul sunt egale atunci fractia este echiunitara;
      - daca numaratorul este mai mic decat numitorul fractia este subunitara.
    </pre>
  </text>
  <symbol name="nr" type="integer"> random(1,20,0);</symbol>
  <symbol name="num" type="integer"> random(1,20,0);</symbol>
  <symbol name="f" type="fractie"> fractieAleatoare2(v("nr"),v("num"));
</symbol>
  <symbol name="raspuns" type="string"> v("f").clasifica();</symbol>
</scenario>

```

Fig.5.1. The Scenario for the fractions classification AGLO

In the theory section, it is presented for the student the part of the theory that is applied in the exercise, namely, the classification of fractions in proper, equal to the unit, or improper. This part is constituted only by static text.

Theory

Pentru fractia m/n avem:

- daca $m > n$ fractia este supraunitara;
- daca $m = n$ fractia este echiunitara;
- altfel fractia este subunitara.

Question

Determinati ce fel de fractie este $15/6$.

Answers

Correct! Evaluate

Feedbacks

Ati avut de determinat daca o fractie este unitara, supraunitara sau subunitara.
 In acest caz fractia este supraunitara.

Fig.5.2. An instance of the fractions classification exercise

The following section contains the actual question, in which the student receives the fraction to be classified. After the instantiation, the symbols are replaced with their values and the student is given a new exercise. In the example in Figure 5.2 the fraction to be classified is 15/6.

The last section is the feedback. For this exercise, feedback is made only by a text, which summarizes the idea of what the student should do, namely, he had to determine if a fraction is proper, equal to the unit, or improper.

5.1.2. Common Divisor AGLO

The second exercise refers to the notion of a common divisor. Although it is a notion that seems to be unrelated to the fractions, it is important to know it. This notion will be used instinctively by the student to solve most of the exercises that need computation of the largest common divisor of the counter and denominator. It is the number by which each fraction is simplified to reach the irreducible form.

The name section is made up of the sentence in which the essence of the AGLO is given, with how much you can simplify a fraction to become an irreducible one.

```
<scenario>
  <text>
    Se generează aleator o fracție ordinară de exemplu 8/20.
    Se calculează cel mai mare divizor comun al numărătorului și numitorului.
    Se returnează divizorul comun, adică 4.
  </text>
  <symbol name="nr" type="integer"> random(1,20,0);</symbol>
  <symbol name="num" type="integer"> random(1,20,0);</symbol>
  <symbol name="f" type="fracție"> fracțieAleatoare2(v("nr"),v("num"));
</symbol>
  <symbol name="divizor" type="integer"> v("f").divizor();</symbol>
</scenario>
```

Fig.5.3. The scenario for the common divisor AGLO

The scenario includes two parts, one in which the workflow of the exercise is explained applied on an example and one in which the symbols are declared and initialized. Two numbers are randomly generated that will represent the numerator and denominator of the fraction. We chose to generate these numbers in the range [1; 20]. The next symbol is a fractional object formed using the two generated numbers. As can be seen in Figure 5.3, the common divisor is returned by a function, since its value will also be needed in subsequent exercises.

The theory section contains a text that explains that a fraction becomes irreducible if it is simplified with the largest common divisor of the counter and denominator.

The question section includes the interrogation: With how much you can simplify the fraction $\frac{\langle \text{value name} = \text{"nr"} \rangle}{\langle \text{value name} = \text{"num"} \rangle}$ to become an irreducible fraction? As one can notice, the static text is embedded with the symbols to be replaced during instantiation with the random values assigned to them

in the scenario section. At each instantiation the two symbols nr and num are replaced with randomly generated numbers in the scenario.

The expected answer to this exercise is simple, a number. It represents the number that is the bigger common divisor of the counter and the denominator.

The screenshot displays a user interface for a fractions exercise. It is divided into four colored sections: a green 'Theory' section with text explaining simplification; a red 'Question' section with the task 'Determinati cu cat se poate simplifica fractia 8/20 pentru a deveni fractie ireductibila.'; a blue 'Answers' section where the user has entered '2' and received a 'Wrong!' message with an 'Evaluate' button; and a yellow 'Feedbacks' section providing a hint: 'Ai avut de calculat cel mai mare divizor comun al numaratorului si numitorului. In acest caz acesta este 4'.

Fig.5.4. An instance of the fractions classification exercise

The feedback section contains a sentence in which the essence of the exercise is accentuated. The student is explained that he had to calculate the bigger common divisor of the count and the denominator. We chose this feedback to emphasize the importance of knowing this mathematical notion because of its importance in the following exercises.

5.1.3. Amplification and Simplification of a Fraction

AGLO

The next two exercises target the following specific skill: Calculation of a fraction equivalent to a given fraction, by amplification or simplification.

Thus the third exercise has the name: Amplification of a fraction.

The scenario section describes how exercise works. It randomly generates a fraction and a number between 1 and 100. It returns a fraction obtained by amplifying the fraction with the previously generated number.

The symbols used in this exercise are:

- nr - the counter of the fraction;
- num - the denominator of the fraction;
- $amplific$ - the number with which the fraction will be amplified;
- f - the actual fraction;

- *raspuns* - which is the answer, namely the amplified fraction.

The first three symbols are randomly generated and based on them the other symbols are initialized.

In the theory section, we defined the mathematical notion of fraction amplifying. To amplify a fraction means to multiply the numerator and the denominator by a given number.

The question section contains an interactive question. The student receives a fraction and a number to amplify this fraction. The number with which it is amplified is limited from 2 to 10 because the idea of amplifying, not having complicated calculations, is important.

The screenshot displays a web-based exercise interface for fraction amplification, organized into four distinct sections:

- Theory:** A green header section containing the text: "A amplifica o fractie inseamna a ii inmulti atat numaratorul cat si numitorul cu un numar dat."
- Question:** A red header section containing the text: "Amplificati 3/20 cu numarul 10."
- Answers:** A blue header section containing a text input field with the value "30/200" and the feedback text "Correct!" below it.
- Feedbacks:** A yellow header section containing the text: "Trebuie sa inmultesti atat numaratorul cat si numitorul cu 10."

Fig.5.5. An instance of the fractions amplification exercise

The answer is the amplified fraction. Figure 5.5 shows an example in which the answer is 30/200, because the fraction 3/20 had to be amplified by 10.

The last section is the feedback. As can be seen in Figure 5.5 in the feedback section, regardless of whether the student's answer is correct or not, the statement reminds the student that he had to multiply the denominator and the numerator by the same number.

In the next exercise, we approached the notion of simplifying fractions. The AGLO name is given by the purpose of the exercise, namely: Simplification of an ordinary fraction.

In the scenario section, we describe how this exercise was thought. A random fraction is randomly generated. It calculates the largest common divisor of the count and denominator. It returns the irreducible fraction obtained from the simplification. If the generated fraction was irreducible then the initial fraction is returned.

To implement this scenario, the following symbols are required:

- the counter and the denominator of the fraction which are randomly generated;
- the common divisor that is computed using the function *divisor()*;
- the counter and denominator of the simplified fraction.

In this exercise the randomly generated values are the counter and the denominator, the other symbol values are calculated based on them.

In the theory section, we have presented both what an irreducible fraction means - whose denominator and counter are prime numbers among themselves - and what it means to simplify a fraction - to divide both the counter and the denominator by the same given number.

```
<question>
  <text>
    Simplificati fractia <value name="nr"/><value name="num"/>astfel
    incat sa obtineti o fractie ireductibila.
  </text>
</question>
```

Fig.5.6. The question section for the simplification of an ordinary fraction AGLO

To create a question that fits the scenario, the statement is made from both static and dynamic text, as can be seen in Figure 5.6. The dynamic part consists of the counter and the denominator of the fraction.

Theory

A simplifica o fractie inseamna a ii imparti atat numaratorul cat si numitorul cu acelasi numar (cel mai mare divizor comun al numaratorului si numitorului).
O fractie este ireductibila daca numaratorul si numitorul sunt doua numere prime intre ele.

Question

Simplificati fractia 8/4 astfel incat sa obtineti o fractie ireductibila.

Answers

4/2

Wrong!

Feedbacks

Fractia trebuia adusa la forma sa ireductibila, fiind simplificata cu 4.

Fig.5.7. An instance of the fractions simplification AGLO with a wrong answer

The answer contains a simplified fraction. This is obtained by dividing the denominator and the numerator by their greatest common divisor.

The feedback contains the idea of the exercise, that is, it is desired to bring the fraction to the irreducible form, and the number with which the fraction must be simplified to reach the desired result. If the student gives a wrong answer, feedback will help him correct his mistake.

An example of the wrong answer is shown in Figure 5.7. The student receives not the correct answer without explanations, but the instructions on what to do.

5.1.4. Comparing Fractions AGLOs

The fifth and sixth AGLO relates to the comparison of fractions. This is also a basic mathematical notion that should be correctly implemented in the mathematical vocabulary of students.

The Comparison of Two Ordinary Fractions AGLO

The fifth AGLO name section is the Comparison of two ordinary fractions.

The scenario section also includes a part consisting of a static text in which the exercise is presented. Two fractions are randomly generated. The two fractions are compared and the sign that should be put between them is returned as follows:

- if the first one is bigger, the ">" sign is returned,
- if equal, returns the "=" sign is returned,
- if the first one is lower, the "<" sign is returned.

```
<scenario>
  <text>
Se genereaza aleator doua fractii ordinare 2/3 si 1/4.
Se compara 2*4 cu 3*1 si se returneaza semnul >.
  </text>
  <symbol name="nr1" type="integer"> random(10,30,0); </symbol>
  <symbol name="num1" type="integer"> random(10,30,0); </symbol>
  <symbol name="nr2" type="integer"> random(10,30,0); </symbol>
  <symbol name="num2" type="integer"> random(10,30,0); </symbol>
  <symbol name="f1" type="fractie">
fractieAleatoare2 (v("nr1"),v("num1")); </symbol>
  <symbol name="f2" type="fractie">
fractieAleatoare2 (v("nr2"),v("num2")); </symbol>
  <symbol name="rezultat" type="string">v("f1").compara(v("f2"));
</symbol>
</scenario>
```

Fig.5.8.The scenario for comparing ordinary fractions AGLO

As can be seen in this exercise, four values, two counters, and two denominators are randomly generated, as two fractions are needed to make a comparison. Each random number is in the range of 10 to 30.

The method *comparare()* was created so it can be used for ordinary fractions as well as for decimal fractions. The fraction from the current object is compared with the fraction that receives it as a parameter. The *rezultat* symbol is the method output and it contains one of the signs representing the answer.

For this object, the feedback section is made only from the static text. It underlines the importance of knowing the theoretical formula.

The Comparison of Two Periodical Mixed Fractions AGLO

The sixth AGLO name section is a Comparison of two periodical mixed fractions.

The scenario section includes a part consisting of a static text in which the exercise is presented and the initialization of the necessary symbols. Two periodical mixed fractions are randomly generated.

```

<scenario>
  <text>
  Se generează aleator doua fracții periodice mixte 3,1(4) și 3,12(5).
  Se transforma în fracții ordinare 283/90 și 2813/900, și se compară conform
  regulii fracțiilor ordinare, adică se compară 283*900 cu 90*2813 și se
  returnează semnul > .
  </text>
  <symbol name="pi" type="integer"> random(10,20,0);</symbol>
  <symbol name="pzn1" type="integer"> random(2,45,0);</symbol>
  <symbol name="pzp1" type="integer"> random(7,20,0);</symbol>
  <symbol name="f1" type="fracție">
  fracțieAleatoare3(v("pi"),v("pzn1"),v("pzp1")); </symbol>
  <symbol name="pzn2" type="integer"> random(2,15,0); </symbol>
  <symbol name="pzp2" type="integer"> random(7,30,0); </symbol>
  <symbol name="f2" type="fracție">
  fracțieAleatoare3(v("pi"),v("pzn2"),v("pzp2")); </symbol>
  <symbol name="rezultat" type="string">v("f1").comparare(v("f2"));
</symbol>
</scenario>

```

Fig.5.9. The scenario for comparing periodical fractions AGLO

In fact, in the background, the method *comparare()* transforms the periodic decimal fractions into ordinary fractions, but the student uses for comparison a more visual method. The whole parts of the two decimal fractions are compared first if they are different, is bigger the fraction with the whole part bigger. If the whole parts are equal, the student compares digit by digit the decimal parts. The comparison stops at the first pair of distinct numbers because at a periodic fraction the fractional part has an infinite number of digits. Thus, the bigger fraction is whose decimal is bigger.

In the scenario, it is explained how the object works, and in the theory section, it is explained to the student how he can compare the fractions by visual method, without making calculations or transformations.

As can be seen in this exercise, the whole parts are equal, so the student must apply the complete algorithm. We chose to put the whole equal because otherwise the problem is reduced to comparing two integers.

The question asks the student to compare the two randomly generated fractions.

The feedback section underlines the importance of knowing how to compare the decimal fractions without making calculations or transformations.

5.1.5. Fraction Addition AGLOs

The following objects deal with fractional operations. The seventh AGLO refers to the addition of the fractions having the same denominator. This is a very simple exercise, but very important from a pedagogical point of view. It is very important to implement this notion in the student's image: two fractions can only be added if they have the same denominator.

The Addition of Two Fractions with the Same Denominator AGLO

The first part of the scenario explains in natural language how to solve the addition of fractions. The addition is further exemplified by a static example. Two ordinary fractions are randomly generated that have the same denominator 7/15 and 2/15. Calculate the numerator, $(7+2)/15$. Simplify the result by obtaining an irreducible fraction, 3/5.

```
<scenario>
  <text>
  Se genereaza aleator doua fractii care au acelasi numitor 7/15 si 2/15.
  Se calculeaza numatorul 7+2, si se obtine fractia suma 9/15.
  Se simplifica rezultatul obtinandu-se o fractie ireductibila: 3/5.
  </text>
  <symbol name="nr1" type="integer">random(1,20,0);</symbol>
  <symbol name="nr2" type="integer">random(1,20,0);</symbol>
  <symbol name="num" type="integer">random(1,30,0);</symbol>
  <symbol name="f" type="fractie">
  fractieAleatoare2(v("nr1")+v("nr2"),v("num")); </symbol>
  <symbol name="raspuns" type="string">
  v("f").numerator/ v("f").divizor()+"/"+v("f").numitor/v("f").divizor();
  </symbol>
</scenario>
```

Fig.5.10. The scenario section for adding two fractions with common denominator

In the scenario section three numbers are randomly generated, two counts, and the common denominator. The fraction that is initialized based on these values is the sum of the fractions. It has the denominator as the common denominator and as the count the sum of the two counts. The answer is the irreducible form of this fraction. To obtain this form, the common divisor method is called.

The theory section presents the theoretical part as a static text: To add two fractions that have the same denominator:

- the two fractions counts are added, this will represent the count of the sum fraction;

- the common denominator will be the denominator of the sum fraction.

The result obtained is simplified to obtain an irreducible fraction.

The question section is composed of both static and dynamic text. The dynamic text is represented by the values of the two generated fractions. Figure 5.11 is represented an example in which fractions $14/52$ and $30/52$ must be added.

Theory

Pentru a aduna două fracții care au același numitor:

- se adună numărătorii celor două fracții, acesta va reprezenta numărătorul fracției suma
- numitorul comun va fi numitorul fracției sumă.

Rezultatul obținut este simplificat pentru a obține o fracție ireductibilă.

Question

Adunați fracțiile:

$14/52$ și $30/52$

Answers

$22/26$

Wrong!

Feedbacks

Fracția corect obținută este $11/13$

Fig.5.11. An instance of the fractions addition AGLO with a wrong answer

The feedback section is giving the student the correct fraction to be obtained. We chose this feedback because it is important if the student gave the wrong answer to see where he made the mistake. In the example above the answer is wrong because fraction $22/26$ is not irreducible, the correct answer is $11/13$.

The Addition of Two Fractions with Different Denominators AGLO

The eighth exercise refers to the addition of two fractions with different denominators. The name of AGLO is the Addition of two common fractions with different denominators.

```

<scenario>
  <text>
    Se genereaza aleator doua fractii ordinare 2/3 si 1/4.
    Se calculeza numarul si numitorul, (2*4+1*3)/(3*4).
    Se simplifica rezultatul obtinandu-se o fractie ireductibila.
  </text>
  <symbol name="nr1" type="integer">random(1,20,0);</symbol>
  <symbol name="nr2" type="integer">random(1,20,0);</symbol>
  <symbol name="num1" type="integer">random(1,30,0);</symbol>
  <symbol name="num2" type="integer">random(1,30,0);</symbol>
  <symbol name="f1" type="fractie">
    fractieAleatoare2 (v("nr1"),v("num1"));</symbol>
  <symbol name="f2" type="fractie">
    fractieAleatoare2 (v("nr2"),v("num2"));</symbol>
  <symbol name="f1" type="fractie"> v("f1").adunare(v("f2"));</symbol>
  <symbol name="raspuns" type="string">
    v("f").numerator/ v("f").divizor()+"/"+v("f").numitor/v("f").divizor();
  </symbol>
</scenario>

```

Fig.5.12. The scenario for the addition of two common fractions AGLO

In the scenario section, Figure 5.12, four numbers are randomly generated, two counts, and two denominators. The counts are generated between 1 and 30, and the denominators are generated between 1 and 20. Three fractional objects are then initialized. The first two are the two fractions obtained from the four randomly generated numbers, and the third is the result returned by the sum method. This method calculates the sum of two fractions. The final result is the irreducible form of the sum fraction.

Just as in the case of adding with the same denominator, the question and feedback sections are composed of both static and dynamic text. In the question section, the dynamic part is made up of the two fractions to be adding, and in the feedback section, it is made up of the fraction to be obtained. The static text is made up of the explanations necessary to make the statements.

Figure 5.13 shows an implementation of the addition of two fractions with different denominators, namely, the addition of fractions 7/10 and 7/2.

The theory section presents the theoretical part as a static text. To gather two fractions, they must be brought to the same denominator. What is to be done is similar to the previous exercise. The sum has as its denominator the common denominator and, as a numerator, the sum of the numerators of the two fractions.

The question section offers the student the two fractions to be added. The result obtained is simplified to an irreducible fraction, namely, 21/5.

The feedback section is composed of a statement that contains the correct fraction to be obtained, 21/5.

Question
Adunati fractiile: $7/10 + 7/2$
Answers
21/5
Correct!
Feedbacks
Fractia corect obtinuta este 21/5

Fig.5. 13. An implementation of the addition of two common fractions AGLO

The addition of two mixed periodic decimal fractions AGLO

The next AGLO refers to the addition of two mixed periodic decimal fractions.

```

<scenario>
  <text>
    Se genereaza aleator doua fractii ordinare 2,7(1) si 1,4(6).
    Se transforma in fractii ordinare si se calculeza suma fractiilor.
    Se simplifica rezultatul obtinandu-se o fractie ireductibila.
  </text>
  <symbol name="pi1" type="integer">random(1,9,0);</symbol>
  <symbol name="pzn1" type="integer">random(6,14,0);</symbol>
  <symbol name="pzp1" type="integer">random(1,5,0);</symbol>
  <symbol name="f1" type="fractie"> fractieAleatoare3
  (v("pi1"),v("pzn1"),v("pzp1"));</symbol>
  <symbol name="pi2" type="integer">random(1,9,0);</symbol>
  <symbol name="pzn2" type="integer">random(1,5,0);</symbol>
  <symbol name="pzp2" type="integer">random(6,14,0);</symbol>
  <symbol name="f2" type="fractie"> fractieAleatoare3
  (v("pi2"),v("pzn2"),v("pzp2"));</symbol>
  <symbol name="f" type="fractie"> v("f1").adunare(v("f2"));
</symbol>
  <symbol name="raspuns" type="string">
  v("f").numerator/v("f").divizor()+"/"+v("f").numitor/v("f").divizor();
</symbol>
</scenario>

```

Fig.5.14. The scenario section for adding two mixed periodic decimal fractions

In the scenario section, see Figure 5.14, six numbers are randomly generated, two whole parts, two decimal non-periodical parts, and two periodical parts. Three fractional objects are then initialized. The first two are the two fractions obtained from the six randomly generated numbers, and the third is the result returned by the sum method.

This exercise was built so that it can be reused for any type of fraction.

The entire fraction consists of a single-digit number to obtain accessible exercises. The decimal parts are, however, chosen from different intervals to give the student varied examples.

The theory section explains to the student the steps to be taken to add two mixed periodic decimal fractions. To add two mixed periodic fractions, they must be transformed into ordinary fractions, brought then to the same denominator. The sum has as its denominator the common denominator and, as a numerator, the sum of the numerators of the two fractions. The result obtained is simplified to obtain an irreducible fraction.

Theory

Pentru a aduna două fracții periodice mixte, ele trebuie transformate în fracții ordinare, apoi aduse la același numitor. Suma are ca numitor numitorul comun și, ca numărător, suma numărătorilor celor două fracții. Rezultatul obținut este simplificat pentru a obține o fracție ireductibilă.

Question

Adunati fractiile:

$5,8(4) + 1,2(7)$

Answers

641/90

Correct!

Evaluate

Feedbacks

Fractia corect obtinuta este 641/90

Fig.5.15. An implementation of adding two mixed periodic decimal fractions AGLO

An example of the implementation of the addition of two mixed periodic decimal fractions is in Figure 5.15. In the question section, are represented the two fractions to be added, namely, $5,8(4)$ and $1,2(7)$.

The answer is represented by the irreducible fraction $641/90$ obtained after performing the addition and simplification operations.

In the feedback section, the student receives regardless of the answer given by the fraction that represents the correct answer.

5.1.6. Fraction Subtraction AGLO

The tenth exercise is aimed at fractions subtraction. This AGLO addresses the idea of subtracting two common fractions. From a mathematical point of view, these fractions may subtract if they have the same denominator.

The name of AGLO is the Subtraction of two common fractions with different denominators.

The first part of the scenario explains in natural language how to solve the subtraction of two fractions with different denominators and is exemplified by a static example.

The screenshot displays a web-based learning interface for a fraction subtraction exercise. It is organized into several colored sections:

- Theory (Green header):** Contains the text: "Pentru a efectua diferenta a doua fractii, acestea trebuie sa aiba acelasi numitor. Diferenta este o fractie a carei numitor este numitorul comun, iar numaratorul este diferenta numaratorilor. Rezultatul obtinut se simplifica obtinandu-se o fractie ireductibila."
- Question (Red header):** Contains the instruction "Efectuati diferenta fractiilor:" followed by a text input field containing the expression $11/4 - 16/20$.
- Answers (Blue header):** Contains a text input field with the answer $39/20$ and a blue "Evaluate" button.
- Feedbacks (Yellow header):** Contains a text box with the feedback message: "Fractia ce trebuia sa o obtineti este: 39/20".

Fig.5.16. An implementation of the subtraction of two common fractions AGLO

In the scenario section four numbers are randomly generated, two counts, and two denominators. The four numbers are in the range 1 20, to give the student easy calculations.

Three fractional objects are then initialized. The first two are the two fractions obtained from the four randomly generated numbers, and the third is the result returned by the subtraction method. This method calculates the subtraction of two fractions. The final result is the irreducible form of the subtraction fraction.

The theory section presents the theoretical part as a static text, Figure 5.16. To make the difference between the two fractions, they must have the same denominator. The difference is a fraction whose denominator is the common denominator, and the numerator is the difference of the numerators. The result obtained is simplified by obtaining an irreducible fraction.

The question presents to the student the fractions to be subtracted, in our example, $11/4$ and $16/20$.

The feedback section is giving the student the correct fraction to be obtained, namely, the $39/20$ fraction.

5.1.7. Fraction Multiplication AGLO

The eleventh and the twelfth exercise are aimed at fractions multiplication. From a mathematical point of view, the resulted fraction is obtained by multiplying the counts, respectively the denominators among them.

The Multiplication of Two Common Fractions AGLO

The name of the eleventh AGLO is the Multiplication of two common fractions.

The first part of the scenario explains in natural language how to solve the requirement and is exemplified by a static example.

In the scenario section four numbers are randomly generated, two counts, and two denominators. The four numbers are in the same range between 1 and 20. Thus avoiding heavy calculations, the emphasis is on the notion that he must acquire.

```

<symbol name="nr1" type="integer">random(1,20,0);</symbol>
<symbol name="num1" type="integer">random(1,20,0);</symbol>
<symbol name="nr2" type="integer">random(1,20,0);</symbol>
<symbol name="num2" type="integer">random(1,20,0);</symbol>
<symbol name="f1" type="fractie">
fractieAleatoare2(v("nr1"),v("num1")); </symbol>
<symbol name="f2" type="fractie">
fractieAleatoare2(v("nr2"),v("num2")); </symbol>
<symbol name="f" type="fractie"> v("f1").inmultire(v("f2"));
</symbol>
<symbol name="raspuns" type="string">
v("f").numarator/v("f").divizor()+"/"+v("f").numitor/v("f").divizor();
</symbol>

```

Fig.5.17. The symbols for the multiplication of two common fractions AGLO

Three fractional objects are then initialized. The first two are the two fractions obtained from the four randomly generated numbers, and the third is the result returned by the multiplication method. This method calculates the multiplication of

two fractions. The final result is the irreducible form of the multiplication fraction. These symbols are illustrated in Figure 5.17.

In terms of structure, the multiplication of two common fractions AGLO is similar to the addition and the subtraction of two common fractions AGLOs.

The theory section presents the theoretical part as a static text. The multiplication of two fractions is a fraction that has as the numerator the multiplication of numerators and as the denominator the multiplication of denominators. The final result is the irreducible fraction obtained by simplifying the product.

The screenshot displays a user interface for a fraction multiplication exercise. It is organized into several colored sections: a green 'Theory' section with explanatory text, a red 'Question' section with the task 'Efectuati produsul fractiilor: 15/10 si 14/11', a blue 'Answers' section containing an input field with the value '210/110', a 'Wrong!' message, and an 'Evaluate' button. Below this is a yellow 'Feedbacks' section with a grey box indicating the correct answer: 'Fractia corect obtinuta este: 21/11'.

Fig.5.18. An implementation of the multiplying of two ordinary fractions AGLO

The question section presents a statement that contains the fractions to be multiplied. In Figure 5.18, the fractions to be multiplied are $15/10$ and $14/11$.

In the previous figure, it is given an example of the implementation of the multiplying of two fractions AGLO with a wrong answer. As can be seen, the feedback section is giving the student the correct fraction to be obtained.

The Multiplication of Two Mixed Periodic Decimal Fractions AGLO

The name of the twelfth AGLO is the Multiplication of two mixed periodic decimal fractions.

In this case, in the scenario section, six numbers are randomly generated two whole parts, two decimal non-periodical parts and, two periodical parts. In this case, the whole parts are restricted to one-digit numbers, the non-periodic decimal parts are digits smaller than 6, and the periodic decimal parts are in the range $[6; 14]$. These restrictions are also imposed in order to make the calculation not difficult.

To multiply two periodic decimal numbers, they must first be transformed into ordinary fractions. For this, three fractional objects are then initialized. The first two are the two fractions obtained from the six randomly generated numbers, and the third is the result returned by the multiplying method.

Theory

Produsul a doua fractii zecimale se obtine transformandu-le in fractii ordinare si efectuand produsul acestora. Produsul a doua fractii ordinare este o fractie ce are ca numarator produsul numaratorilor, iar ca numitor produsul numitorilor. Rezultatul final este fractia ireductibila ce se obtine prin simplificarea produsului.

Question

Efectuati produsul fractiilor: $8,7(3)$ si $4,3(13)$

Answers

55937/1485

[Evaluate](#)

Correct!

Feedbacks

Fractia corect obtinuta este: 55937/1485

Fig.5.19. An implementation of the multiplying of two decimal fractions AGLO

As in the previous AGLO, the theory section presents how to perform the multiplication of two periodic decimal fractions.

The question section presents to the student the fractions to be multiplied. In Figure 5.19 is represented an implementation of this AGLO in which the student is asked to multiply the decimal fractions $8,7(3)$ and $4,3(13)$.

The feedback section is giving the student the correct fraction to be obtained, namely $55937/1485$.

5.1.8. Fraction Power AGLO

The next AGLO addresses the rise to power of a fraction. The name of AGLO is the Power of a fraction.

The first part of the scenario explains in natural language how to solve the requirement and is exemplified by a static example. In the second part of the scenario section six numbers are randomly generated, an option, a count, a denominator, an

integer part, a periodical part, and a number that represents the power to which the fraction is raised.

Two fractions are generated:

- an ordinary fraction that has the denominator and the count generated previously;
- a simple periodic decimal fraction.

The option can have a value of 1 or 0. Depending on its value, the question and the answer are generated. If the option has value 1, the power base will be an ordinary fraction, and if the option has the value 0, the power base will be a simple periodic fraction. Fraction power is implemented as a method that returns the power fraction.

```
<scenario>
...
<symbol name="optiune" type="integer">random(0,1,0)</symbol>
<symbol name="nr" type="integer">random(1,10,0)</symbol>
<symbol name="num" type="integer">random(1,10,0)</symbol>
<symbol name="pi" type="integer">random(1,5,0)</symbol>
<symbol name="pz" type="integer">random(1,9,0)</symbol>
<symbol name="putere" type="integer">random(2,5,0)</symbol>
<symbol name="f1" type="fractie">
fractieAleatoare2(v("nr"),v("num"));</symbol>
<symbol name="f2" type="fractie">
fractieAleatoare3(v("pi"),"",v("pz"));</symbol>
<symbol name="var1" type="string">v("nr")+"/"+v("num");</symbol>
<symbol name="var2" type="string">v("pi")+","+(v("pz")+");</symbol>
<symbol name="intrebare" type="string"> v("optiune") ? v("var1") :
v("var2");</symbol>
<symbol name="raspuns" type="string"> v("optiune") ?
v("f1").laPuterea(v("putere")) : v("f2").laPuterea(v("putere"));</symbol>
</scenario>
```

Fig.5.20. The scenario of the fraction power AGLO

Because power fraction is obtained by repeatedly multiplying the fraction as many times as the exponent looks like, we have restricted the exponent to be generated in the range 1 to 6, and denominator and counter fraction between 1 and 10. In the case of the periodic fraction, we further restricted the whole part to avoid getting difficult numbers to calculate.

The theory section presents the theoretical part as a static text. The power of a number is obtained by multiplying the number with itself as many times as equal to the exponent.

The question section is composed of both static and dynamic text. It tells the student what the base number is and what the exponent is to calculate the power.

The feedback section is composed of

- a static part that explains how to obtain the power of a fraction;
- a dynamic text, giving the student the correct fraction to be obtained.

Figure 5.21 shows two implementations of the AGLO, one in which an ordinary fraction $2/10$ is required to rise to power 2 and one in which a simple decimal fraction $1,(7)$ is required to be raised to power 4. In both cases, the result is an ordinary fraction.

The figure displays two side-by-side screenshots of an AGLO (Assessment of Learning Objectives) interface for fraction power. Each screenshot is divided into four colored sections: Theory (green), Question (red), Answers (blue), and Feedbacks (yellow).

Left Screenshot:

- Theory:** Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de ori egal cu exponentul.
- Question:** Ridicati fractia $\frac{2}{10}$ la puterea 2
- Answers:** The input field contains $\frac{4}{100}$. Below it, the text "Correct!" is displayed, and an "Evaluate" button is visible.
- Feedbacks:** Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de ori egal cu exponentul. Trebuia sa obtineti fractia $\frac{4}{100}$.

Right Screenshot:

- Theory:** Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de ori egal cu exponentul.
- Question:** Ridicati fractia $1,(7)$ la puterea 4
- Answers:** The input field contains $\frac{65536}{6561}$. Below it, the text "Correct!" is displayed, and an "Evaluate" button is visible.
- Feedbacks:** Puterea unui numar se obtine inmultind numarul cu el insusi de un numar de ori egal cu exponentul. Trebuia sa obtineti fractia $\frac{65536}{6561}$.

Fig.5.21. Two implementations of the fraction power AGLO

5.1.9. Fraction Division AGLO

The fourteenth and the fifteenth AGLO are aimed at the fractions division.

From a mathematical point of view, the resulted fraction is obtained by multiplying the first fraction with the inverse of the second fraction. If the fractions are given as decimal numbers, the first step is to transform them into ordinary fractions and then divide them.

The Division of Two Common Fractions AGLO

The name of the fourteenth AGLO is the Division of two common fractions.

The first part of the scenario explains in natural language the steps to be made to perform the fraction division and the mathematical operation is exemplified by a static example.

In the scenario section four numbers are randomly generated, two counts and two denominators, in the range $[1; 20]$. Three fractional objects are then initialized. The first two are the two fractions obtained from the four randomly generated numbers, and the third is the result returned by the division method. The final result is the irreducible form of the resultant fraction. These symbols are illustrated in Figure 5.22.


```

<symbol name="nr1" type="integer">random(1,80,0);</symbol>
<symbol name="num1" type="integer">random(1,80,0);</symbol>
<symbol name="nr2" type="integer">random(1,80,0);</symbol>
<symbol name="num2" type="integer">random(1,80,0);</symbol>
<symbol name="f1" type="fractie">fractieAleatoare2(v("nr1"),v("num1"));
</symbol>
<symbol name="f2" type="fractie">
fractieAleatoare2(v("nr2"),v("num2")); </symbol>
<symbol name="f" type="fractie"> v("f1").impartire(v("f2")); </symbol>
<symbol name="raspuns" type="string">
v("f").numarator/v("f").divizor()+"/"+v("f").numitor/v("f").divizor();
</symbol>

```

Fig.5.22. The symbols for the division of two common fractions AGLO

The theory section presents the theoretical part as a static text. In this section, we explain the fact that the division of two fractions is done by multiplying the first fraction by the inverse of the second.

Theory

A impartii doua fractii inseamna a inmulti prima fractie cu inversul celei de-a doua. Rezultatul final este o fractie ireductibila obtinuta din simplificarea fractiei cat.

Question

Efectuati catul fractiilor:

$12/19 : 10/2$

Answers

$24/190$

Wrong!

Feedbacks

Fractia corect obtinuta este: 12/95

Fig.5.23. An implementation of the fraction division AGLO

The question section contains a dynamic part represented by the variables randomly generated previously. Figure 5.23 shows an implementation of this AGLO in which the student is asked to divide the ordinary fractions $12/19$ and $10/2$.

We chose the implementation in Figure 5.23 to have a wrong answer to see that the feedback section gives the student both the correct result to be obtained and a recapitulation of the notion.

The Division of Two Mixed Periodic Decimal Fractions AGLO

The name of the fifteenth AGLO is the Division of two mixed periodic decimal fractions.

In this case in the scenario section six numbers are randomly generated, two whole parts, two decimal non-periodical parts, and two periodical parts. In this case, the whole parts are restricted to one-digit numbers, the non-periodic decimal parts are digits smaller than 6, and the periodic decimal parts are in the range [6; 14]. These restrictions are imposed in order not to make the calculation difficult.

Three fractional objects are then initialized. The first two are the two fractions obtained from the six randomly generated numbers, and the third is the result returned by the division method.

Theory

Catul a doua fractii zecimale se obtine transformandu-le in fractii ordinare si efectuand impartirea acestora. A imparti doua fractii ordinare inseamna a inmulti prima fractie cu inversul celei de-a doua. Rezultatul final este o fractie ireductibila obtinuta din simplificarea fractiei cat.

Question

Efectuati catul fractiilor:

2,7(1) si 9,5(8)

Answers

244/863

Correct!

Evaluate

Feedbacks

Fractia corect obtinuta este: 244/863

Fig.5.24. An implementation of the fraction division AGLO

The theory section contains the instructions necessary to perform the exercise, namely the decimal fractions are first transformed into simple fractions, and then they are performed. The student is also told that the end result will be an irreducible fraction.

The question section presents to the student the fractions to be divided. Figure 5.24 shows an implementation of this AGLO in which the student is asked to calculate the quotient of fractions 2.7 (1) and 9.5 (8). The student must transform the two

fractions into ordinary fractions, namely $244/90$ and $863/90$. The result obtained after the division is $244/863$.

The feedback section is giving the student the correct fraction to be obtained to help him if he was wrong.

5.1.10. Finding a Percentage of a Whole AGLO

The next AGLO aims to find out some percent of a whole. Although it is an elementary notion, students need to acquire this properly. It is also a very common notion in daily extracurricular activities.

```
<scenario>
<text>
Se genereaza aleator un procent si un numar intreg.
Se calculeaza procentul din acel numar (procent*numar)/100.
</text>
<symbol name="numar" type="integer">random(1,500,0);</symbol>
<symbol name="procent" type="integer">random(1,100,0);</symbol>
<symbol name="rezultat" type="float">v("procent")*v("numar")/100;
</symbol>
</scenario>
```

Fig.5.25.The scenario for finding a percentage of a whole AGLO

Theory
p% din x reprezinta numarul $(p \cdot x)/100$
Question
Calculati 31% din numarul 289, cu doua zecimale exacte.
Answers
89.59
Correct!

Fig.5.26. An implementation of the finding a percentage of a whole AGLO

As can be seen in Figure 5.25, the scenario section comprises both a part of static text, which is transcribed how the item works and a part of generating random numbers for the required variables. A percentage and an integer are needed. The percentage is chosen randomly from the range $[1; 100]$, and the integer is chosen

randomly from the range [1; 500]. It is calculated the percentage of that number by the formula (percentage * number) / 100. In the case of this exercise, there was no need to implement any specific method. The notion is not difficult to implement, but the concept is very important to be well understood by students.

The theory section presents the theoretical part, in mathematical language, as a static text: $p\%$ of x represents the number $(p * x) / 100$.

The questions section is composed of both text and dynamic values represented by the percentage and the number from which it is calculated. In the example of implementation from Figure 5.26, it is required to calculate 31% of the number 289.

The feedback section is composed only of the correct number to be obtained, in the above implementation the correct answer being the decimal number 89.59.

5.1.11. Transforming a Decimal Fraction AGLO

The seventeenth AGLO refers to the transforming of a decimal fraction into an ordinary fraction.

```
<scenario>
  <text>
  Se genereaza aleator o fractie zecimala simpla 8,75.
  Se calculeza numaratorul si numitorul, (8*100+75)/100.
  </text>
  <symbol name="pi" type="integer">random(1,20,0);</symbol>
  <symbol name="pzn" type="integer">random(1,200,0);</symbol>
  <symbol name="f" type="fractie">fractieAleatoare3(v("pi"),v("pzn"), "");
</symbol>
  <symbol name="numitor" type="integer">v("f").numitor;</symbol>
  <symbol name="numarator" type="integer">v("f").numarator;</symbol>
</scenario>
```

Fig.5.27.The scenario for transforming a simple decimal fraction into an ordinary fraction AGLO

In the scenario section, two numbers representing the whole part and the decimal part are generated randomly. With the help of these numbers, the *fractie* object is constructed. A whole part is a random number between 1 and 20. A decimal part is a random number between 1 and 200. The restriction on the decimal part determines the obtaining of fractions that can have as the denominator the numbers 10, 100, or 1000.

The counter and denominator of this fraction will be the answer to the following question.

The theory section presents the theoretical part as a static text. To transform a simple decimal fraction into an ordinary one, the following are performed:

- the denominator is put 1 followed by so many zeros how many digits has the simple decimal fraction after the comma

- the counter is calculated without taking into account the punctuation marks.

The question section is composed of:

- static text – the statement: Convert the following simple decimal fraction into an ordinary fraction:

- dynamic text - the fraction to be transformed.

Theory

Pentru a transforma o fractie zecimala simpla intr-una ordinara se realizeaza urmatoarele:

- la numitor se pune 1 urmat de atatea zerouri cate cifre are fractie zecimala simpla dupa virgula;
- la numarator se pune numarul fara sa tinem cont de semnele de punctuatie.

Question

Transformati in fractie ordinara urmatoarea fractie zecimala simpla:

12,139

Answers

12139/1000

You got 10 out of 10!

Fig.5.28. An implementation of transforming a decimal fraction into a simple fraction AGLO

An implementation of the AGLO is shown in Figure 5.28. In the feedback, the student gets the correct fraction to be obtained.

5.1.12. Transforming a Periodic Decimal Fraction AGLO

The eighteenth AGLO refers to transforming a periodic decimal fraction into a simple fraction.

In the scenario section, the whole part and the periodic decimal part are generated randomly. When the fraction builder is called the non-periodic decimal part is null.

The theory section presents the theoretical part. To transform a simple periodic decimal fraction into an ordinary fraction, the following are performed:

- the denominator, there are so many 9 digits, how many digits are in the period;
- the initial number is placed on the counter without taking into account the punctuation marks from which the non-periodic part is subtracted.

The question section also includes a dynamic part, in which at each implementation it is randomly generated another fraction. An exercise to transform a periodic decimal fraction, namely 13.(12) into an ordinary fraction is shown in Figure 5.30.

```

<scenario>
  <text>
    Se genereaza aleator o fractie zecimala periodica, ex. 12.(34).
    Se calculeza numaratorul si numitorul,  $12 + \frac{34}{99} = \frac{1222}{99}$ 
  </text>
  <symbol name="pi" type="integer">random(10,20,0);</symbol>
  <symbol name="pz" type="integer">random(10,20,0);</symbol>
  <symbol name="f" type="fractie">fractieAleatoare3(v("pi"), "", v("pz"));
</symbol>
  <symbol name="numitor" type="integer">v("f").numitor;</symbol>
  <symbol name="numarator" type="integer">v("f").numarator;</symbol>
  <symbol name="raspuns" type="string">v("numarator")+"/"+v("numitor");
</symbol>
</scenario>

```

Fig.5.29.The scenario for transforming a simple decimal fraction into an ordinary fraction AGLO

Theory

Pentru a transforma o fractie zecimala periodica simpla intr-o fractie ordinara se realizeaza urmatoarele:

- la numitor se pun atatia de 9 cate cifre sunt in perioada
- la numarator se pune numarul fara sa tinem cont de semnele de punctuatie din care se scade partea neperiodica.

Daca este posibil se simplifica fractia astfel obtinuta.

Question

Transformati in fractie ordinara urmatoarea fractie periodica simpla:

13, (12)

Answers

433/33

Evaluate

Correct!

Feedbacks

Se obtine fractia 1299/99
Care simplificata devine 433/33

Fig.5.30. An implementation of transforming a simple periodic decimal fraction into an ordinary fraction AGLO

In the feedback section, the student receives the fraction obtained by transformation and if necessary, the correct fraction to be obtained by simplification.

In the example above, the initial fraction is $1299/99$. By simplification, the answer is obtained, namely $433/33$.

5.1.13. Transforming a Mixed Periodic Decimal Fraction AGLO

The fifteenth AGLO aims the transforming of a mixed periodic decimal fraction into a simple fraction.

In the scenario section, the whole part, the non-periodical decimal part, and the periodic decimal part are generated randomly.

```
<scenario>
  <text>
  Se genereaza aleator o fractie zecimala periodica mixta, ex. 12.34(56) .
  Se calculeaza numaratorul si numitorul,  $12+(3456-34)/9900 = 12 + 3422/9900 = 122222/9900 = 61111/4950$ 
  </text>
  <symbol name="pi" type="integer">random(10,20,0);</symbol>
  <symbol name="pzn" type="integer">random(10,20,0);</symbol>
  <symbol name="pzp" type="integer"> random(10,20,0);</symbol>
  <symbol name="f" type="fractie">
  fractieAleatoare3(v("pi"),v("pzn"),v("pzp"));</symbol>
  <symbol name="numitor" type="integer">v("f").numitor;</symbol>
  <symbol name="numarator" type="integer">v("f").numarator;</symbol>
  <symbol name="raspuns" type="string">
  v("f").numitor+"/"+v("f").numitor</symbol>
  <symbol name="r" type="string"> v("f").numarator/v("f").divizor()
  +"/"+v("f").numitor/v("f").divizor();</symbol>
  <symbol name="fb" type="string"> v("f").divizor()==1 ?
  " Fractie ireductibila." : " Care simplificata devine "+v("r");
  </symbol>
</scenario>
```

Fig.5.31. The scenario for transforming a mixed periodic decimal fraction into a simple fraction AGLO

The theory section presents the theory, to convert a mixed decimal fraction into an ordinary fraction, the following are performed:

- to the denominator, we will put as many digits of 9 as many digits are in the period and as many digits of zero as many digits are after the comma, but not in the period

- the numerator is the initial number without taking into account the punctuation marks from which the non-periodic part is subtracted.

A static example is also given, namely $12.34(56) = 122222/9900 = 61111/4950$.

The question section contains the mixed periodic decimal fraction, and the feedback section is giving the student the correct simple fraction to be obtained, and where appropriate the simplified fraction.

5.1.14. Introduction of the Whole into a Fraction AGLO

The twentieth AGLO aims the introduction of the whole into a fraction. For this, a subunit fraction and a whole are generated randomly.

The scenario section contains the explanation of what you want to calculate in the exercise and a static example. Then the variables to be initialized are given. For the fraction to be subunit, we chose to generate the counter in the range of 1 to 10 and the denominator in the range of 11 to 20. To have accessible calculations, we chose the whole to be in the range [2; 20].

The theory section presents the theoretical part. To introduce the whole into the fraction we perform the following:

- for the new count, we multiply the integer by the denominator and add it to the old count;

- the denominator remains the same.

Figure 5.32 illustrates an implementation of the AGLO in which it is required to enter in the fraction 4 integers and 3/11.

The screenshot displays a user interface for an AGLO (Assessment Goal Learning Object) titled 'Introduction of the whole into a fraction'. It is divided into several sections:

- Theory:** A green header section containing the text: 'In cazul introducerii intregilor infractie se obtine o fractie noua astfel: - numaratorul se obtine inmultind intregii cu numitorul si adunand la vechiul numaratorul, - numitorul ramane acelasi.'
- Question:** A red header section with the instruction: 'Introduceti in fractie 4 intregi si 3/11.'
- Answers:** A blue header section containing a text input field with the value '47/11' entered.
- Feedback:** A yellow header section containing a feedback message: '47/11 47/11 OK' and 'You got 10 out of 10!'.

Fig.5.32. An implementation of introduction of the wholes into a fraction AGLO

The numerator and denominator of the fraction obtained from the calculations is the answer that is expected to be given. In the feedback section, the student receives the correct answer, and this is compared with the answer given by him.

5.1.15. Extracting the Whole from a Fraction AGLO

The seventeenth exercise aims to extract the whole from a fraction.

Removing the whole from an ordinary fraction is a method of the fraction object.

In the descriptive part of the scenario, that contains the static text, an example is given to illustrate what is desired to be learned through this exercise. The wholes from a fraction represent the quotient of dividing the counter by the denominator.

To be mathematically correct and to be as useful as possible, we have chosen that the interval in which the denominator is generated is between 1 and 9, and the one in which the counter is generated is between 10 and 20. In this way, we made sure that every time there is at least one integer, and the operation to be solved has a medium level.

```
<scenario>
  <text>
Se genereazaaleator o fractie ordinara, de exemplu 20/8.
Se calculeza catul impartirii 20/8=2.
  </text>
  <symbol name="nr" type="integer"> random(10,20,0);</symbol>
  <symbol name="num" type="integer"> random(1,9,0);</symbol>
  <symbol name="f" type="fractie"> fractieAleatoare2(v("nr"),v("num"));
</symbol>
  <symbol name="raspuns" type="string"> v("f").intregii();</symbol>
</scenario>
```

Fig.5.33.The scenario for extracting the wholes from a fraction AGLO

The theory section presents the theoretical part as a static text. In the example part it is presented to the student that in fraction 37/11 there are 3 integers.

Theory

Se imparte numaratorul la numitor. Catul reprezinta intregii care ies din fractie, iar restul va fi noul numarator.
De exemplu:
Intregii din fractia 37/11 sunt 3.

Question

Scoateti intregii din fractia 19/7.

Answers

2

Fig.5.34.The theory, question and answers sections for extracting the wholes from a fraction AGLO

An ordinary fraction is randomly generated. In the question section, the student is asked to remove the whole from the respective fraction.

The student calculates the quotient of the division between the count and the denominator. It represents the whole of that fraction, that is, the answer.

At each implementation of this AGLO, the student receives an ordinary fraction from which he must remove the whole. In Figure 5.34, an implementation of this AGLO is presented in which the student is asked to extract the wholes from the fraction $19/7$. In this example, the answer is two.

In the feedback section, it is emphasized to the student that the answer is obtained by dividing the counter by the denominator.

5.2. AGLOs for intervals, equations and inequations involving absolute values and inequalities

Intervals, equations and inequalities are among the basic notions in algebra. Their correct understanding can be achieved by practicing the applications that integrate them.

Regarding the intervals, we created AGLOs to help the student practice all the operations: intersection, union and difference.

5.2.1. Intervals AGLO

The first AGLO targets the intersection of two intervals.

In the scenario section, four integers are randomly generated that will represent the bounds of the two intervals. We chose the left bound from both intervals to be generated in the same interval $[-20, 20]$, and the right bound to be calculated based on the left one where we add a randomly generated number between 1 and 20, one different for each interval. Thus the student has the possibility to obtain both the case when the intersection is empty and the one in which the intersection is not empty. In addition, this way we made sure that the intervals are mathematically correct.

In the theory section it is explained that the intersection of two intervals A and B is an interval that contains numbers which are in both of the intervals. It is the set of numbers that are in A and B [61]. If there are no such numbers then the intersection is the empty set. The student receives an example for each case, see Figure 5.35.

The question section comprises the two randomly generated intervals. In the example given above, the intervals were generated $[4;7]$ and $[-5;9]$.

In the answer section, the interval given by us, namely, $[4;7]$ is evaluated. We have chosen to offer a correct answer, so the grade received is maximum.

In the feedback section, the student is reminded that the intersection contains the common elements of the two intervals. It also includes the correct answer to be given. We chose the left bound from both intervals to be generated in the same interval $[-15, 15]$, and the right bound to be calculated based on the left one where

we add a randomly generated number between 1 and 30, one different for each interval.

The next AGLO aims the union of two intervals.

In the scenario section, four integers are randomly generated, like the previous example. That integers will represent the bounds of the two intervals.

In the theory section it is presented that the union of two intervals, A and B, is an interval that contains all elements from A and B [61]. An example is also represented, namely the union of $[-4;8]$ and $[7;9]$ intervals.

In the question section, the question is formulated based on the randomly generated values in the scenario. In Figure 5.36 the question asked is to calculate the union of the intervals $[-3;21]$ and $[-14;9]$.

The screenshot displays a user interface for an Adaptive Generation Learning Object (AGLO) focused on interval intersection. It is divided into several sections:

- Theory:** A green header section containing a definition: "The intersection of two intervals A and B is an interval that contains the common elements of intervals A and B." Below this are two examples: $[-2 ; 3] \cap [5 ; 7] = \text{empty}$ and $[-4 ; 7] \cap [5 ; 11] = [5 ; 7]$.
- Question:** A red header section containing the problem: $[4 ; 7] \cap [-5 ; 9]$.
- Answers:** A blue header section containing a text input field with the value $[4 ; 7]$. Below the input is a feedback message: "You got 10 out of 10!". To the right of the input is a blue button labeled "Evaluate".
- Feedbacks:** A yellow header section containing a detailed explanation: "The intersection of two intervals contains the common elements of intervals, in this case $[4 ; 7]$." Below this is a list of input elements: $[4$, $]$, 4 , OK , $;$, $;$, OK , 7 , $]$, 7 , OK .

Fig.5.35. An implementation of the intersection of two intervals AGLO

In the answer section we chose to provide the correct answer, namely the interval $[-14;21]$. It is evaluated and noted accordingly when activating the *Evaluate* button. In the example in Figure 5.36 we gave the correct answer.

The correct answer is provided in the feedback section. The student also receives some recommendations regarding the union calculation. In the case of disjoint intervals, the meeting does not consist of a single interval, otherwise the left end point of the meeting is the smaller of the two numbers representing the left end point of the two initial intervals, and the right end point of the meeting is the higher number. between the two correct endpoints of the initial intervals.

Theory

The reunion of two intervals, A and B, is an interval that contains all elements from A and B.
 Example:
 $[-4 ; 8] \cup [7 ; 9] = [-4 ; 9]$

Question

The reunion of intervals $[-3 ; 21] \cup [-14 ; 9]$ is:

Answers

[Evaluate](#)

You got 10 out of 10!

Feedbacks

The reunion of intervals is $[-14 ; 21]$ interval.
 In the case of disjoint intervals, the reunion does not consist of a single interval, and otherwise the left endpoint of the reunion is the smaller of the two numbers representing the left endpoint of the two initial intervals, and the right endpoint of the reunion is the bigger number of the two right endpoints of the initial intervals.

```
[-14 [-14 OK
; ; OK
21] 21] OK
```

Fig.5.36. An implementation of the union of two intervals AGLO

The third AGLO refers to the difference of intervals. In this case, two intervals are randomly generated in the scenario section. The left bound of the first interval is generated between -10 and 10, and the left bound of the second interval is generated between -20 and 20. The right bounds are calculated based on the left ones where we add a randomly generated number between 1 and 20, one different for each interval.

In the theory section it is explained to the student that the difference of the intervals A and B is the interval formed by those elements that are found in A but are not found in B. Also presented are two examples, one in which the intersection is a

compact interval and one in which intersection consists of the meeting of two intervals. It is also specified that the capital letter U will be used instead of the reunion sign in the answer (see Figure 5.37).

The question is asked in the next section. In the example above the student must calculate the difference between the intervals $[8,20]$ and $[18,24]$.

Theory

Diferenta intervalelor A si B este intervalul formatat din acele elemente care se gasesc in A dar nu se gasesc in B.
 $[-4 ; 6] - [5 ; 8] = [-4 ; 5]$
 Observatie: Daca e cazul, pentru semnul de reuniune folositi litera U (mare).
 $[-20 ; 10] - [-2 ; 7] = [-20 ; -2) \cup (7 ; 10]$

Question

Diferenta intervalelor
 $[8 ; 20] - [18 ; 24]$
 este intervalul ...

Answers

$[8 ; 18)$

You got 10 out of 10! Evaluate

Feedbacks

Elementele din primul interval care nu se gasesc si in al doilea interval: $[8 ; 18)$

$[8$ OK
 $;$ OK
 $18)$ OK

Fig.5.37. An implementation of the difference of two intervals AGLO

In the feedback section the student receives the observation that the difference includes the elements from the first interval that are not found in the second interval. He is also provided with the correct answer which is compared to the answer given by him.

5.2.2. Equations involving absolute values AGLO

The following AGLO aims to solve an equation involving absolute values.

The equation has the form $|ax + b| = c$. In this case, in the scenario section, three numbers a , b and c are randomly generated. The randomly generated numbers are:

- a , the x coefficient, chosen from the numbers 2, 4, 5, 8, 10;
- the term b from the interval $[1; 100]$;

- the term c from the interval $[1; 100]$.

We chose these values because a it will be a nominator in the solution and we want to get integers or finite decimal numbers. Thus we maintain an average level for the generated exercises, and we ensure the existence of accessible numbers, the emphasis falling on the way in which the exercise is performed more than on calculations.

A rezolva o ecuatie inseamna a ii determina multimea solutiilor.
 Pentru ecuatia $|x + 2| = 7$
 avem solutiile:
 -9 5

Question

Pentru ecuatia $|2*x + 6| = 14$
 avem solutiile:

Answers

-10 4

Evaluate

You got 10 out of 10!

Feedbacks

Aveti de rezolvat doua ecuatii:
 $2*x + 6 = -14$
 $2*x + 6 = 14$

-10 -10 OK
 4 4 OK

Fig.5.38. An implementation of equations involving absolute values AGLO

In the theory section, the student is reminded what it means to solve an equation, namely to determine the set of solutions. An example is presented to him, namely solving the equation $|x + 2| = 7$. Here the student can also see the form in which he has to write the answer, namely the two solutions in ascending order separated by space. In the case of the example, the solution consists of numbers -9 and 5.

The question statement is formulated in the question section. Figure 5.38 shows an example of this AGLO in which it is required to determine the solutions of the equation $|2x+6|=14$.

The feedback section shows the student how to solve the equation with the module, namely solving two equations $a \cdot x + b = -c$ and $a \cdot x + b = c$. In the case of the above example the solutions are -10 and 4.

5.2.3. Inequations AGLO

Regarding the inequations, the first AGLO aims at solving the inequations involving absolute values.

An inequation involving absolute values has the form $|ax + b| \leq c$. In the scenario section, three numbers are randomly generated:

- a , a number from the set $\{2,4,5,8,10\}$ that will represent the coefficient of x ;
- b , an integer in $[-50; +50]$ and
- c , an integer in $[0; 200]$.

Theory

To solve an inequation means to determine its set of solutions.
 For the inequation: $|2x + 5| \leq 7$,
 the set of solutions is the interval:
 $[-6; 1]$

Question

Solve the following inequation on the set of real numbers:
 $|10x - 23| \leq 89$

Answers

$[-6.6; 11.2]$

👁 Evaluate

You got 10 out of 10!

Feedbacks

The correct interval is $[-6.6; 11.2]$.
 The inequation is equivalently written:
 $-89 \leq 10x - 23 \leq 89$

$[-6.6; 11.2]$ OK
 ; ; OK

Fig.5.39. An implementation of Inequations involving absolute values AGLO

Based on these numbers, we calculate the interval according to the mathematical formula $\left[\frac{-c-b}{a}; \frac{c-b}{a}\right]$. We have chosen these restrictions for a because it will be divisor and we want to get finite decimal numbers.

Furthermore, an inequation of form $|ax + b| \leq c$ will become a double inequality of form $-c \leq ax + b \leq c$. For this reason we chose the term c to be a positive number, to obtain a true mathematical inequality.

The theory section reminds us that solving an inequation means determining its set of solutions. Also represented is a static example, namely solving the inequation $|2x + 5| \leq 7$, which has as solution the interval $[-6; 1]$. This example also shows the student that the solution that is expected to be written is a bounded interval.

Theory

A rezolva o inecuatie inseamna a ii determina multimea solutiilor.
 Pentru ecuatie: $3*x + 5 \leq 8$
 Multimea solutiilor este: $(-\infty ; 1]$

Question

Rezolvati urmatoarea inecuatie:
 $2*x + 18 \leq 61$

Answers

$(-\infty ; 21.5]$

Evaluate

You got 10 out of 10!

Feedbacks

$(-\infty ; 21.5]$.
 $(-\infty (-\infty$ OK
 $; ;$ OK
 $21.5] 21.5]$ OK

Fig.5.40. An implementation of inequation with an interval left unbounded and right-bounded as solution AGLO

Figure 5.39 shows an implementation of this AGLO. In the question section in this case, it is required to solve the inequation $|10x - 23| \leq 89$. The question statement is made with the help of randomly generated numbers in the scenario section.

In the answer section we chose to write the correct answer, namely the interval $[-6.6; 11.2]$. Following the evaluation, the grade obtained is maximum.

In the feedback section, the student receives an indication of how the inequatio turns into double inequality, so that it can be solved. Then his answer is compared to the correct answer.

The following AGLO aims at solving an inequation that has as a solution an interval left-unbounded and right-bounded. For this exercise in the scenario section, three integers are randomly generated:

- the coefficient of x restricted to the set $\{2,4,5,8,10,20\}$;
- the term b in the range $[1; 100]$;
- the term c in the range $[1; 100]$.

In this way we can say that there is the possibility to generate 60,000 different examples of inequations.

In the theory part, the student is reminded of the notion and is also presented with an example, namely solving the inequation $3x + 5 \leq 8$. In the case of the example, the interval that represents the solution is $(-\infty; 1]$. Through this example the student also sees the way in which the answer must be written correctly.

The question contains the inequation made using randomly generated numbers in the scenario. Figure 5.40 shows an implementation of this AGLO, the inequation that needs to be solved being $2x + 18 \leq 61$.

The answer given by us in the example presented in Figure 5.40 is the correct one, namely the interval $(-\infty; 21.5]$. The correct answer automatically generated grade 10.

In the feedback section, the answer given by the student is compared element by element with the correct answer calculated in the scenario.

The following AGLO aims at solving inequations of the form $-a * x + b \leq c$ that has as solution an interval left-bounded and right-unbounded.

For this exercise in the scenario section, three integers are randomly generated:

- the coefficient of x restricted to the set $\{2, 4, 5, 8, 10, 20\}$;
- the term b in the range $[1; 100]$; and
- the term c in the range $[1; 100]$.

Since the meaning of the inequality is identical to that of the previous example, the difference is obtained by transforming the coefficient of x into a negative number. In this way the same type of inequality has another solution. In fact, the student must know that dividing the inequality with a negative number, its meaning is reversed. Thus the solution is an interval left-bounded and right-unbounded.

In the theory part, the student is reminded of the notion and is also presented with an example, namely solving the inequation $-3x + 5 \leq 8$. In the case of the example, the interval that represents the solution is $[-1; +\infty)$.

The question contains the inequation made using randomly generated numbers in the scenario. Figure 5.41 shows an implementation of this AGLO, the inequation that needs to be solved being $-10x + 85 \leq 95$.

The answer given by us in the above example is the correct one, namely the interval $[-1; +\infty)$. This automatically generate the grade 10.

In the feedback section, the answer given by the student is compared with the correct answer calculated in the scenario.

Theory

A rezolva o inecuatie inseamna a ii determina multimea solutiilor.
 Pentru ecuatia: $-3*x + 5 \leq 8$
 Multimea solutiilor este: $[-1 ; +infinite)$

Question

Rezolvati urmatoarea inecuatie:
 $-10*x + 85 \leq 95$

Answers

You got 10 out of 10!

Feedbacks

$[-1 ; +infinite)$.
 $[-1 [-1 OK$
 $; ; OK$
 $+infinite) +infinite) OK$

Fig.5.41. An implementation of inequations of the form $-a * x + b \leq c$ AGLO

The following AGLO that aims at solving inequations of the form $a * x + b \geq c$. In this case the unbounded side and the bounded side, will depend on the randomly generated values. Thus, this exercise is a generalization of the previous two.

For this exercise in the scenario section, three integers are randomly generated:

- the coefficient of x , from the set $\{-2, 2, 4, -5, 5, 8, -10, 10, -20, 20\}$;
- the term b in the interval $[-50; 50]$;
- the term c in the interval $[-50; 100]$.

In the theory part, the student is reminded of the notion and is also presented with an example, namely solving the inequation $-3x + 5 \geq 8$, see Figure 5.42. In the example case, the interval that represents the solution is $(-infinite; -1]$.

Theory

A rezolva o inecuatie inseamna a ii determina multimea solutiilor.
 Pentru ecuatie: $-3x + 5 \geq 8$
 Multimea solutiilor este: $(-\infty ; -1]$

Question

Rezolvati urmatoarea inecuatie:
 $-10x + 9 \geq 82$

Answers

$(-\infty ; -7.3]$

You got 10 out of 10! [Evaluate](#)

Feedbacks

$(-\infty ; -7.3]$.

$(-\infty (-\infty$ OK
 $; ;$ OK
 $-7.3] -7.3]$ OK

Fig.5.42. An implementation of inequations of the form $a * x + b \geq c$ AGLO

In the example from Figure 5.42, in the question section the student is asked to solve the inequation $-10x + 9 \geq 82$. In this case, the interval that represents the solution is $(-\infty; -7.3]$.

The correct answer given will generate the maximum grade, namely 10.

In the feedback section, the student is given the correct answer, with which the answer given by him is compared. Thus, if he made a mistake, the student can see which of the sides of the interval is wrong.

The next AGLO aims at solving double inequations.

For this exercise in the scenario section, three integers are randomly generated, the coefficient of x restricted to the set $\{-2, 2, 4, -5, 5, 8, -10, 10, -20, 20\}$, and the terms b and c in the range $[1; 100]$.

In the theory part, the student is reminded of the notion and is also presented with an example, namely solving the inequation $-4 \leq 3x + 5 \leq 8$, see Figure 5.43. In the example case, the interval that represents the solution is $[-3; 1]$.

In the example from Figure 5.43, in the question section the student is asked to solve the inequation $12 \leq -2x - 21 \leq 28$. In this case, the interval that represents the solution is $[-24.5; -16.5]$.

Theory

A rezolva o inecuatie inseamna a ii determina multimea solutiilor.
 Ecuatia: $-4 \leq 3x + 5 \leq 8$
 Are ca solutie intervalul: $[-3 ; 1]$

Question

Rezolvati urmatoarea inecuatie:
 $12 \leq -2x - 21 \leq 28$

Answers

$[-24.5 ; -16.5]$

You got 10 out of 10!

Feedbacks

Se fac calcule astfel incat intre cele doua semne sa ramana doar x: $-24.5 \leq x \leq -16.5$
 Intervalul corect este: $[-24.5 ; -16.5]$.

$[-24.5 [-24.5$
 $; ;$ OK
 $-16.5] -16.5]$ OK

Fig.5.43. An implementation of double inequations AGLO

In the feedback section, the student is reminded of the method by which this type of inequality is calculated, namely calculations are made until only x remains between the two signs. Next, the answer given by the student is compared with the correct answer expected.

5.3. AGLOs for algebraic calculation formulas

A very important topic in eighth grade math is the correct use of abbreviated calculation formulas. These formulas must be very well mastered by students as they will be used in solving several types of exercises.

5.3.1. Application of abbreviated calculation formulas AGLO

The first AGLO aims at the formulas $(a \pm b)^2 = a^2 \pm 2ab + b^2$.

For the first exercise in the scenario section, an integer is randomly generated in the interval $[2;20]$, and an option that can be 0 or 1. These intervals ensures the generation of 38 different variants. The option is to choose which of the two formulas will be used, namely the square of an amount or the square of a difference.

The student only has to apply the formula. Figure 5.45 shows an example in which the number 10 was randomly generated.

In the feedback section is represented the correct application of the formula, namely $x^2 + 20x + 100$.

Theory	Theory
$(x + 1)^2 = x^2 + 2x + 1$ $(x - 1)^2 = x^2 - 2x + 1$	$(x - 1)(x + 1) = x^2 - 1$
Question	Question
Calculati $(x - 10)^2$	Calculati $(x - 9)(x + 9)$
Answers	Answers
$x^2 - 20x + 100$	$x^2 - 81$
You got 10 out of 10!	You got 10 out of 10!
Feedbacks	Feedbacks
Rezultatul este: $x^2 - 20x + 100$.	$x^2 - 81$.
x^2 x^2 OK $-$ $-$ OK $20x$ $20x$ OK $+$ $+$ OK 100 100 OK	x^2 x^2 OK $-$ $-$ OK 81 81 OK

Fig.5.44. An implementation of abbreviated calculation formulas AGLOs

The next AGLO aims at the formula $(a + b)(a - b) = a^2 - b^2$.

For this exercise in the scenario section, is randomly generated an integer in the interval $[2;20]$. In Figure 5.44 it is represented an implementation of this AGLO in witch the randomly generated number is 9. And in this case, solving the exercise is simple, namely the student must apply the appropriate formula.

5.3.2. Decomposition into factors AGLO

This exercises are the revers from the one in the previous section. The first AGLO aims at the formulas $a^2 \pm 2ab + b^2 = (a \pm b)^2$.

For this exercise in the scenario section, an integer is randomly generated in the interval $[2;20]$, and an option that can be 0 or 1. The option has the role of choosing the sign that differs between the two targeted formulas. In Figure 5.46 it is represented an implementation of this AGLO in witch the randomly generated number is 4, and the result is the square of a difference.

<p>Theory</p> <p>Formulele:</p> $x^2 + 2*x*y + y^2 = (x + y)^2$ $x^2 - 2*x*y + y^2 = (x - y)^2$ <p>Un exemplu de utilizare:</p> $x^2 + 2*x + 1 = (x + 1)^2$	<p>Theory</p> <p>Formula:</p> $x^2 - y^2 = (x - y)(x + y)$ <p>Exemplu:</p> $x^2 - 1 = (x - 1)(x + 1)$
<p>Question</p> <p>Restrangeti dupa formula : $x^2 - 8*x + 16$</p>	<p>Question</p> <p>Restrangeti dupa formula : $x^2 - 64$</p>
<p>Answers</p> <p><input type="text" value="(x - 4)^2"/></p> <p>You got 10 out of 10!</p>	<p>Answers</p> <p><input type="text" value="(x - 8)(x + 8)"/></p> <p>You got 10 out of 10! Evaluate</p>
<p>Feedbacks</p> <p>Se obtine patrutul: $(x - 4)^2$.</p> <p>(x (x OK - - OK 4)^2 4)^2 OK</p>	<p>Feedbacks</p> <p>$(x - 8)(x + 8)$.</p> <p>(x (x OK - - OK 8)(x 8)(x OK + + OK</p>

Fig.5.45. An implementation of decompositions into factors AGLOs

The next AGLO aims at the formula $a^2 - b^2 = (a + b)(a - b)$. And in this case the notion is quite simple, but it must be well understood, and this can be achieved by practicing. In this case the student must know that the difference of two squares can be narrowed as a product between the difference and the sum of the two roots.

For this exercise in the scenario section, is randomly generated an integer in the interval $[2;20]$. In Figure 5.45 it is represented an implementation of this AGLO in witch the randomly generated number is 8.

5.4. Summary

In this chapter we presented the AGLO that aims several Arithmetic notions.

For working with fractions we have made a number of twenty-one AGLOs. These exercises cover the specific competencies related to the fractions required by the current curriculum in Romania for fifth grade students.

Thus, we managed to make thirty-four AGLOs aimed at middle school mathematics, namely the arithmetic part. We can conclude that in this chapter we presented that the AGLO model can be applied to the notions of arithmetic in middle school.

For eighth grade arithmetic, we made a number of thirteen exercises. These exercises cover specific competencies related to the first two chapters of arithmetic required by the current curriculum in Romania for eighth grade students. These notions refer to operations with intervals, solving equations and inequalities, as well as to some abbreviated calculation formulas.

We can conclude that AGLOs can be folded very well on arithmetic formulas.

6. AGLOS FOR INFORMATION TECHNOLOGY DISCIPLINES

6.1. Models for Data Structures and Algorithms Discipline

The table is a fundamentally structured type because it is defined directly in the language, with hardware support for indexing.

One of the most common operations, that are performed on the tables, is the search, that is, the verification of the existence of a given element in the table. We have approached three types of searches: linear search, linear search with sentinel, binary search, and search by interpolation.

Another operation performed on the tables is sorting. Regarding the sorting methods, we chose to make AGLOs both with the simple sorting methods, namely insertion sort, selection sort, bubble sort, and with the advanced ones, namely shell sort, Quicksort. These AGLOs refer both to the steps of the algorithms and to notions specific to each one.

Another important notion in terms of structures is the notion of list. We have created AGLOs that target, random or ordered, single-linked linear lists and double-linked linear lists. We aimed at recognizing specific elements and operations such as adding or deleting in the list according to a certain requirement.

6.1.1. Linear search AGLOs

Linear search is a method of determining the smallest index for which the element of the table $tab[i]$ is equal to the searched element x . The elements of the table are compared one by one with x until the equality of $tab[i] = x$ is reached or the end of the table is reached [62].

To deepen these notions, we have chosen to make thirteen applications that address different aspects of the problem. To have all the functions we need we have created the `LinearSearch.js` library. This includes the constructor function and the function of determining the algorithm steps.

The first AGLO approaches the definition of the notion of linear search in a table. For educators, this item illustrates the familiarity of the student with the notion.

An implementation of this AGLO is shown in Figure 6.1. This AGLO offers the student the opportunity to choose the correct answer between two possible variants. Always next to the correct version it will have an incorrect one. The incorrect variant is generated randomly and at each instance, it has a different form.

The correct definition of the linear search is resumed in the feedback section.

Question

Ce rol are algoritmul de cautare liniara?

Answers

Algoritmul de cautare liniara are rolul de a ascunde o valoare anume intr-un tablou la un moment dat pana cand valoarea dorita este gasita.

Algoritmul de cautare liniara are rolul de a gasi o valoare anume intr-un tablou si consta in verificarea fiecărei valori la un moment dat in mod secvential pana cand valoarea dorita este gasita.

Fig.6.1. The question and answer of the linear search definition AGLO

The following AGLO familiarizes the student with the linear search algorithm steps.

```

<scenario>
  <text>
Se genereaza un tablou aleator de n = 6-11 elemente. Se listeaza toti pasii cautarii
liniare.
  </text>
  <symbolname="n"type="integer">random(6,11,0);</symbol>
  <symbolname="i"type="integer">random(0,v("n")-1,0);</symbol>
  <symbolname="tab"type="array">random_array({"nNoOfElements":v("n"),"tabTypes":["
"int"]}).tabElements;</symbol>
  <symbolname="x"type="integer">v("tab")[i];</symbol>
  <symbolname="ls"type="object">linear_search({"array":v("tab"),"element":v("x"),
});</symbol>
  <symbolname="steps"type="string">v("ls").getSteps();</symbol>
</scenario>

```

Fig.6.2. The scenario section of the linear searching algorithm steps

Figure 6.2 illustrates the scenario of the AGLO that aims for the student to know the steps of the linear search algorithm. For the realization of this topic, a table of n integer elements is generated randomly. The number n is between 6 and 11. For the searched element, an element from the generated table is also randomly chosen. We choose to take one of the table elements because we want to verify the understanding of the search steps and the fact that this algorithm stops at the first element found.

The expected answer is to list all the steps of the linear search until the element is found. For the student to know the correct form that the answer should have in the theory part, we gave a static example, as can be seen in Figure 6.3.

In the feedback section, each step of the search is compared item by item with the correct answer. We thought this way of feedback would help the student. If he does not know the correct answer, he can see how far the algorithm went correctly and where the error occurs. In addition, the grade that the student receives is

proportional to the correctness of the answer. I chose that in this situation there are not only the right and wrong options.

Theory	Feedbacks
<p>Daca se da tabloul: <code>tab=[3,6,8,14]</code> si se cauta <code>x=8</code> pasii algoritmului sunt urmatoari:</p> <pre>i=0 tab[0]==8? 3==8? nu i=1 tab[1]==8? 6==8? nu i=2 tab[2]==8? 8==8? da</pre>	<pre>i=0 i=0 OK 0<9 0<9 OK 16==207 16==207 OK ? ? OK nu nu OK i=1 i=1 OK 1<9 1<9 OK 182==207 182==207 OK ? ? OK nu nu OK i=2 i=2 OK 2<9 2<9 OK 291==207 291==207 OK ? ? OK nu nu OK i=3 i=3 OK 3<9 3<9 OK 207==207 207==207 OK ? ? OK da da OK</pre>
<p>Question</p> <p>Daca se da tabloul: <code>tab=[16,182,291,207,22,20,111,82,261]</code> si se cauta <code>x=207</code>,care sunt pasii algoritmului?</p>	
<p>Answers</p> <pre>i=0 0<9 16==207 ? nu i=1 1<9 182==207 ? nu i=2 2<9 291==207 ? nu i=3 3<9 207==207 ? da</pre> <p>You got 10 out of 10!</p>	

Fig.6.3. An implementation of the linear searching algorithm steps.

The next AGLO aims the understanding the working principle of the algorithm. In the theory section, the student receives the necessary notions that are applied in solving the exercise. The linear search algorithm sequentially compares each element of the table with the value to be searched. If it finds a match, it returns the value of the index where that match was found. Otherwise, it is considered that the searched element is not present in the table.

To implement this exercise, several elements located in the range 10 -15, and a table of random numbers of this size are randomly generated. Also, an element from the string is chosen at random, this being the element to look for.

Figure 6.4 shows the implementation of the operating principle of the algorithm. In this case, the student must determine how many comparisons the algorithm executes.

The next AGLO refers to the practical situations in which the linear search algorithm is used.

The student receives three practical situations in which this algorithm is used:

- when we have a list or an array of a certain size;
- when a search in an unordered table is needed;
- when a search is needed in an orderly picture that undergoes frequent changes and its reorganization becomes a burden.

What changes at instantiation is the number of elements of the table, it is chosen in the range 100 - 200.

Theory

Algoritmul de cautare liniara compara secvential fiecare element al tabloului cu valoarea de cautat. Daca gaseste o coincidenta returneaza valoarea indicelui unde s-a gasit acea coincidenta. In caz contrar se considera ca elementul cautat nu este prezent in tablou.

Question

Cate comparatii se executa la cautarea lui $x=297$ in sirul $tab=[175,189,99,69,128,250,47,158,30,97,297]$?

Answers

11

Fig.6.4. The implementation of the operating principle of the linear searching algorithm AGLO

In this situation, all the options are correct, and in the feedback part, the student receives the necessary theoretical explanations to deepen knowledge.

Question

In ce situatii practice este folosita cautarea liniara?

Answers

Cand lista sau tabloul are doar 171 elemente.
 Cand e nevoie de o cautare intr-un tablou neordonat.
 Cand e nevoie de o cautare intr-un tablou ordonat ce sufera modificari frecvente si reorganizarea lui devine o povara.

Correct! Evaluate

Feedbacks

Cautarea liniara se aplica pe structuri indiferent de dimensiunea lor. Cautarea liniara se aplica de regula pe tablouri neordonate, pe cele ordonate exista alte cautari mai performante. Daca tabloul trebuie reordonat de foarte multe ori nu merita aplicarea unor algoritmi de cautare mai avansati, cautarea liniara este solutia cea mai buna in acest caz.

Fig.6.5. An implementation of the practical situation in which the linear search algorithm is used AGLO

In Figure 6.5 it is captured an implementation of the practical situation in which the linear search algorithm is used AGLO with a correct answer given.

In the next applications, the student receives strings of different types of data characters, integers, strings, randomly generated, and he must choose the variants on which the algorithm is applicable.

In the scenario section, are initialized four variables, the number of string elements that are located in the range 10 to 15, and the three tables.

Question

Algoritmul de cautare liniara se poate aplica pe:

Answers

tablou aleator de intregi [46, 18, 262, 261, 234, 20, 9, 128, 208, 203, 100]

tablou aleator de caractere ['a', 'c', 'f', 'g', 'i', 'l', 'o', 'r', 't', 'w', 'z']

tablou aleator de siruri de caractere ["audi", "bmw", "mercedes", "toyota", "lexus", "honda", "acura"]

Correct! Evaluate

Feedbacks

Nu exista un alt algoritm de cautare pentru tablourile neordonate.

Fig.6.6. An implementation of applying the linear search on tables of basic data types AGLO

In Figure 6.6, you can see an implementation of applying the linear search algorithm on a random data table of various types of elements AGLO. The idea highlighted here is that this is an algorithm applicable to unordered strings. This idea is reminded to the student in the feedback section.

The following AGLO refers to the use of the algorithm on ordered tables.

The main idea is that the linear search can be applied on any tables with basic type elements or with structured elements.

In the scenario section, five tables needed to present the desired options:

- a table with ordered scalar elements;
- a table of strings with ordered elements;
- three table with structured elements that are ordered by a string or an integer key.

The elements of these tables are randomly generated in numerical or alphabetical order.

In the question section, the student is asked to choose the tables on which the algorithm can be applied.

In the answer section, the five tables are presented to the student. In Figure 6.7, an implementation of the application on different random data tables is showed. At each instantiation, all five situations appear, but the elements of the tables change.

In the feedback section, regarding the ordered tables, the student is advised to use a binary search for these. Indeed linear search is not performance, but it is useful in situations where the elements of the table are random.

Question

Pe care din tablourile de mai jos se poate aplica cautarea liniara?

Answers

- Pe tabloul [78, 90, 104, 184, 240, 310, 328, 385, 423].
- Pe tabloul ["audi", "bmw", "lexus", "mercedes"].
- Pe tabloul [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }] avand ca si cheie de cautare campul de tip sir de caractere.
- Pe tabloul [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }] avand ca si cheie campul de tip intreg.
- Pe tabloul [{adresa:0x26, nume:"alin", }, {adresa:0x33, nume:"barbu", }, {adresa:0x42, nume:"cezar", }, {adresa:0x53, nume:"dorin", }] avand ca si cheie campul de tip sir de caractere.

Correct!

Evaluate

Fig.6.7. An implementation of the application of linear search on different random data tables AGLO

The following AGLO refers to the result that the algorithm returns. What should be kept in mind is that the linear search algorithm returns the first position on which the searched element is found if it is in the array.

To perform this exercise, in the scenario section, the number of table elements in the range 7 - 12, and the table elements are randomly generated. The element to be searched is chosen randomly from the elements of the string in order not to have the answer: the element does not exist in the table.

In the question section, the student is asked to determine the result of the linear search algorithm applied to the values generated in the scenario. Figure 6.8 shows an implementation of this AGLO in which the student is asked to determine the result of the search for element 250 in the table consisting of elements 68, 250, 205, 177, 47, 158, 60 and 137.

In the answer section, the student is given two different options for each implementation, one is true and one is false. In the case of the example in Figure 6.8. the correct variant is the position of the searched element, namely the index 1.

This idea taken up in the feedback section is that this algorithm returns the first position where it finds this element.

Question

Care este rezultatul algoritmului de cautare liniara in cazul in care sirul este [68, 250, 205, 177, 47, 158, 60, 137] iar elementul cautat este 250.

Answers

- Pozitia elementului cautat, adica indexul 1.
 Chiar elementul cautat 250.
 Correct!

Evaluate

Feedbacks

Cautarea liniara returneaza pozitia elementului gasit 1.
 Cautarea liniara nu trebuie sa returneze valoarea elementului cautat 250, ci pozitia sau indexul acestuia 1.

Fig.6.8. The result of the searching algorithm AGLO

The following AGLO refers to the result that the linear search algorithm returns when is applied on a linked list.

Question

Care este rezultatul algoritmului de cautare liniara in cazul in care lista inlantuita este
 [{"address": "0xe", "value": 75, "next": "0x12"}, {"address": "0x12", "value": 63, "next": "0x1a"}, {"address": "0x1a", "value": 23, "next": "0x1e"}, {"address": "0x1e", "value": 75, "next": "0x26"}, {"address": "0x26", "value": 40, "next": "0x32"}, {"address": "0x32", "value": 63, "next": "0x36"}, {"address": "0x36", "value": 66, "next": "0x3a"}, {"address": "0x3a", "value": 59, "next": "0x3e"}, {"address": "0x3e", "value": 39, "next": "0x46"}, {"address": "0x46", "value": 78, "next": "0x00"}]
 iar elementul cautat este 75?

Answers

- Adresa elementului cautat 0xe.
 Valoarea elementului cautat 75.
 Correct!

Evaluate

Feedbacks

Cautarea liniara returneaza adresa de memorie 0xe a elementului gasit.
 Cautarea liniara nu trebuie sa returneze valoarea elementului cautat 75, ci adresa acestuia 0xe.

Fig.6.9. An implementation of the result of the LINEAR searching algorithm APPLIED on a linked list AGLO

In this AGLO we chose the list to have between 7 and 15 randomly generated elements. In the answer section, the student is given two different options for each

implementation, one is true and represents the address of the element, and the other is false and represents the value of the element.

Figure 6.9 shows an implementation of the linear searching algorithm applied on a linked list. In this case, we chose the correct answer, namely the address of the element. The idea is once again highlighted in the feedback section.

Question

Care din algoritmi de mai jos sunt algoritmi de cautare liniara?

Answers

```
// A1
i=n-1;
cat timp i>=0 si a[i]!=x
  i=i-1;

daca a[i]==x
  atunci
    afiseaza "coincidenta la pozitia "+i
  altfel
    afiseaza "elementul cautat nu exista in tablou"
```

```
// A2
pentru i=n-1; i>=0; i=i-1
daca a[i]==x
  atunci
    afiseaza "coincidenta la pozitia "+i
```

```
// A3
i=n;
executa
  i=i-1
cat timp i>=0 si a[i]!=x
  daca a[i]==x
  atunci
    afiseaza "coincidenta la pozitia "+i
  altfel
    afiseaza "elementul cautat nu exista in tablou"
```

Correct!

Feedbacks

Algoritmul A1 este implementat pe baza instructiunii while (cat timp). Algoritmul A2 este implementat pe baza instructiunii for (pentru). Algoritmul A3 este implementat pe baza instructiunii do while (executa cat timp).

Fig.6.10. An implementation of the recognition of an implementation of linear search AGLO

An implementation of the next AGLO is illustrated in Figure 6.10. In this exercise, the student must recognize several implementation variants of this algorithm in pseudocode. Every time, the student receives three real variants.

Thus are present the variants:

- The A1 algorithm is implemented based on the while instruction;
- The A2 algorithm is implemented based on the for instruction;
- The A3 algorithm is implemented based on the do-while instruction.

What is dynamic in this example are the names of the indices, the table, the length of the table. It is about implementing the algorithm with different repetitive structures. This idea is important because in this way the student observes the equivalence between repetitive instructions.

The next AGLO refers to the recognition of the algorithm applied on a table.

In this exercise, the student receives a program sequence, written in the C programming language, represented in the Figure 6.11. This sequence represents the linear search algorithm applied on a board. At each instant the name of the table changes, the index, and the name of the element to be searched.

Question

Ce algoritm reprezinta codul de mai jos?

```

i=0;
while((i<n-1) && !(e[i]==x))
    i=i+1;

if(e[i]==x)
    printf("coincidenta la pozitia %d",i);
else
    printf("elementul cautat nu exista in tablou");

```

Fig.6.11. The question of the recognition of an implementation of linear search on a table AGLO

The student must recognize both the algorithm and the type of data to which it applies. These are reminded in the feedback section.

The next AGLO refers to the recognition of the linear searching algorithm applied on a table of structures.

Question

Ce algoritm reprezinta codul de mai jos?

```

i=0;
while((i<n-1) && !(strcmp(c[i].titlu,x)==0))
    i=i+1;

if(strcmp(c[i].titlu,x)==0)
    printf("coincidenta la pozitia %d",i);
else
    printf("elementul cautat nu exista in tablou");

```

Answers

- 1. Cautare liniara pe tablou de structuri.
- 2. Cautare liniara pe lista inlantuita.
- 3. Cautare binara pe tablou de structuri.
- 4. Cautare binara pe lista inlantuita.
- 5. Cautare prin interpolare pe tablou de structuri.
- 6. Cautare prin interpolare pe lista inlantuita.

Correct!

[Evaluate](#)

Feedbacks

Se observa ciclul ce porneste de la 0 la n-1 si comparatia care cauta elementul in sir. Cautarea liniara se executa secvential.

Fig.6.12. An implementation of the recognition of an implementation of linear search on a table of structures AGLO

In the question section is a program sequence that represents the linear search on a table of structures with different fields of different types. Randomly during implementation are selected:

- as table_name one of the letters: a, b, c, d, e, f, g, h, i, j;
- as field_name one of the variants: name, surname, cnp, color, brand, model, author, title, diary, average, age, date_of_birth, year_of_birth.

Figure 6.12 represents an implementation of the recognition of the implementation of linear search on a table of structures.

And this time the student must take into account both the characteristics of the linear search, namely the sequential search, and the type of data on which it is applied.

The next AGLO also refers to the recognition of the linear searching algorithm, but this time applied on a linked list.

Question

Ce algoritm reprezinta secventa de cod de mai jos?

```
typedef struct nod
{
    int cheie;
    struct nod *urm;
}NOD;

NOD *prim,*ultim;
NOD *p;
...
p=prim;
while ((p!=NULL) && !(p->cheie==x))
    p=p->urm;

if(p->cheie==x)
    printf("coincidenta la pozitia %d",i);
else
    printf("elementul cautat nu exista in tablou");
```

Answers

- 1. Cautare liniara pe tablou de structuri.
- 2. Cautare liniara pe lista inlantuita.
- 3. Cautare binara pe tabloul de structuri.
- 4. Cautare binara pe lista inlantuita.
- 5. Cautare prin interpolare pe tablou de structuri.
- 6. Cautare prin interpolare pe lista inlantuita.

Correct!

Feedbacks

Secventa de cod reprezinta o cautare liniara aplicata pe o lista inlantuita. Se observa structura NOD si pointerul p ce parcurge lista inlantuita incepand cu primul ei element. Se mai observa si instructiunea de atribuire pentru avansarea la elementul urmator p=p->urm.

Fig.6.13. An implementation of the recognition of an implementation of linear search on a linked list AGLO

In this exercise, the student receives a program sequence in which a linked list is initially defined, and then the linear search algorithm applied to a linked list. The dynamic part consists of the names of the variables.

In this case, in the answer section, the student receives several types of algorithms as variants and he must choose the correct algorithm among them, see Figure 6.13. The code sequence is a linear search applied on a linked list.

In the feedback part, is reminded of the theoretical part that is applied to the correct solution. It is observed the existence of the *NOD* structure, the pointer *p* that goes through the chained list starting with its first element, and the assignment instruction for advancing to the next element $p = p \rightarrow next$.

6.1.2. Linear search with sentinel AGLOs

The linear search algorithm has been improved by a method that involves the use of a sentinel. The table is extended with one more element (the sentinel), which has the value of *x*. Then the linear search method is applied, that is, the element *x* is searched sequentially in the table. The advantage of this method is the simplification of the cycling condition, in the sense that it is no longer necessary to check if the index does not exceed the size of the table because in the table there is at least one element with the looked-up value [62].

For this search, we have made AGLOs which deals with the following aspects: definition of the notion, algorithm steps, operating principle, different use cases for different types of values, as well as the recognition of some implementations for different types of input data.

The first AGLO targets the notion of linear search with a sentinel. It is shown that the search algorithm with sentinel has the role of finding a particular value in a table and that it is an improved variant of the classical linear search. In this exercise, the student has a choice between two statements, one false and one true about the search with sentinel.

The second AGLO tests the student's ability to recognize an implementation in the C programming language of the search algorithm.

In the question section, is presented a sequence of code that represents the linear search with the sentinel. What is dynamic in this exercise is the fact that each implementation randomly chooses the name of the array, index, and search element from several variants.

In the answer section, the student is offered three options from which to choose the correct one, see Figure 6.14.

In the feedback section, the particularities of each search are resumed and underlined. These must be very well understood by the student to be able to use the algorithms correctly. This is a linear search with a sentinel because we can identify: the table is *e*, the counter is *j*, the size of the array is *r*, and the searched element is *x*, and the sentinel has been added at the end of the table. This is not a binary search because we do not have specific indexes for the left and right of the interval that this algorithm applies. In addition, it is not an interpolation search because we do not have specific indexes for the left and right of the interval that this algorithm applies, nor the pivot interpolation formula.

Question

Ce algoritm reprezinta secventa de cod de mai jos ?

```
e[r]=x;
j=0;
while ( !(e[j]==x))
    j=j+1;

if(j<r)
    printf("coincidenta la pozitia %d",j);
else
    printf("elementul cautat nu exista in tablou");
```

Answers

1. Cautare liniara cu fanion
 2. Cautare binara
 3. Cautare prin interpolare
 Evaluate

Correct!

Feedbacks

Este vorba de cautare liniara cu fanion deoarece putem identifica: tabloul este e, contorul este j, dimensiunea tabloului este r, elementul cautat este x, iar fanionul a fost adaugat la sfarsitul tabloului.

Fig.6.14. An implementation of the searching algorithm with sentinel recognition AGLO

The next AGLO refers to the recognition of the position of the sentinel.

To perform this exercise, in the scenario section, a number between 8 and 15 is randomly generated. This number represents the number of elements that the table will have. The elements of the table and the element to be searched are then randomly generated. We chose the intervals in such a way that the search element is not in the table.

In the question section, the student is presented with the table and the element to look for. He is asked to determine the position that the sentinel will occupy.

The student must know that the sentinel is inserted on the last position of the table after its extension with an element, see Figure 6.15. This idea is also emphasized in the feedback section.

The next AGLO refers to the value of the sentinel. In this case, the student receives also a table and a value to look for. He has to say what the value of the sentinel is if the searching algorithm with sentinel is applied to the table. Here the student must know that the sentinel has the value of the element sought.

In the scenario of this exercises it is necessary to randomly generate a number that will represent the number of elements of the table, the elements of the table, and the element to be searched.

In this case, the element to be searched may or may not be in the table, being chosen randomly from the same range as the elements of the table.

Se da tabloul de mai jos:

[29, 69, 61, 59, 50, 39, 42, 16, 48, 51]

in care dorim sa cautam valoarea 62.
Pe ce pozitie trebuie inserat fanionul in cadrul cautarii liniare cu fanion?

Answers

10

You got 10 out of 10! [Evalueaza](#)

Feedbacks

Fanionul se insereaza pe ultima pozitie a tabloului dupa extinderea acestuia cu un element.

Fig.6.15. An implementation of the recognition of the position of the sentinel AGLO

In the question section, the student is presented with the table and the element to look for. He is asked to determine the value of the sentinel.

Se da tabloul de mai jos:

[31, 53, 27, 37, 60, 21, 11, 17, 61, 46, 14, 67, 51, 38]

in care dorim sa cautam valoarea 74.
Care este valoarea fanionului ce trebuie inserat?

Answers

74

You got 10 out of 10! [Evaluati](#)

Feedbacks

Fanionul are chiar valoarea elementului cautat si anume 74.

Fig.6. 16. An implementation of the recognition of the sentinel value AGLO

The student must know that the sentinel has the value of the element sought, see Figure 6.16.

The last AGLO that refers to the search algorithm with sentinel shows the student comparison between this search and the classical linear search.

```

    <symbol name="found" type="boolean"> random(0,1,0)==1 ? true : false;
</symbol>
    <symbol name="n" type="integer"> random(8,12,0); </symbol>
    <symbol name="tab" type="array"> random_array({"nNoOfElements" :
v("n"), "tabTypes" : ["integer"], "nMin":10, "nMax":60}); </symbol>
    <symbol name="i" type="integer"> random(0,v("n")-1,0); </symbol>
    <symbol name="x" type="integer"> v("found") ? v("tab").toArray()[v("i")]
: random(70,130,0); </symbol>
    <symbol name="c2" type="integer"> v("found") ? 2*(v("i")+1) : 2*v("n");
</symbol>
    <symbol name="c" type="integer"> v("found") ? v("i")+1 : v("n"); </symbol>
    <symbol name="raspuns" type="integer"> v("c2") - v("c"); </symbol>

```

Fig.6.17. The symbols from the scenario of the comparison between linear search with sentinel and the classical linear search AGLO

We have chosen in this exercise to include both the case when the element is found in the string and the case when it is not found in the string. The role of the first implemented symbol *found* is to determine which situation will be implemented:

- 0 represents the case when the element is not in the table,
- 1 represents the case where the element is in the table.

The length of the string, which can vary between 8 and 12, the elements of the string, and an index of an element in the string are then randomly generated.

The following variables are generated based on the value of *found*: the element to be searched, and the number of steps for each algorithm.

The answer *raspuns* will be the difference between the numbers of the two algorithms steps.

In the theory section, we chose to emphasize that in the case of the classical linear search at each step, two comparisons are performed, one to find the element and the other to verify that the size of the table is exceeded, see Figure 6.18.

In the question section, the two algorithms written in the C programming language applied to the generated string are represented.

The student should determine how many more comparisons are performed in the case of linear search. There is a formula for determining them. It is worth considering that even if the element exists in the table, the difference of comparisons is significant.

In the feedback section, it is recalled what the two comparisons represent in the case of the classic linear search.

Theory

In cazul cautarii liniare la fiecare pas se executa doua comparatii una pentru gasirea elementului si cealalta pentru verificarea depasirii dimensiunii tabloului.

Question

Se dau doi algoritmi unul de cautare liniara clasic si unul de cautare liniara cu fanion.
Care este diferenta dintre numarul de comparatii in primul respectiv al doilea caz?

<pre>// versiunea clasica, fara fanion int tab[12]={[29, 54, 19, 57, 32, 27, 46, 16, 58, 17, 25, 12]}; int n=12; int x=95; int i=0; while(tab[i]!=x && i<n) { i++; } if(tab[i]==x) { printf("elementul a fost gasit pe pozitia %d",i); } else { printf("elementul nu a fost gasit"); }</pre>	<pre>// versiunea cu fanion int tab[12]={[29, 54, 19, 57, 32, 27, 46, 16, 58, 17, 25, 95]}; int n=12; int x=95; tab[12]=95; int i=0; while(tab[i]!=x) { i++; } if(i==n) { printf("elementul nu a fost gasit"); } else { printf("elementul a fost gasit pe pozitia %d",i); }</pre>
---	---

Answers

[Evaluate](#)

You got 10 out of 10!

Feedbacks

In primul caz la fiecare pas avem de executat cate doua comparatii:
i) una pentru gasirea elementului
ii) si cealalta pentru verificarea depasirii dimensiunii tabloului.
In al doilea caz la fiecare pas avem de executat doar cate o comparatie, dar tabloul trebuie extins cu locatia unde va fi inserat fanionul.

12 12 OK

Fig.6.18. An implementation of the comparison between linear search with sentinel and the classical linear search AGLO

6.1.3. Binary search AGLOs

The binary search algorithm is a search algorithm used to find an element in an ordered list. The principle of this search method consists of the repeated halving of the interval in which the desired element is searched.

We consider that we apply the algorithm on an ordered table of elements named `tab`. It is considered `tab[m]` the element located in the middle of the ordered table. It compares with the searched element:

- if they are equal, the algorithm ends and returns m the position of the element;
- if the searched element is smaller than the algorithm resumes for the elements from the beginning of the table to the middle;
- if the searched element is larger than the algorithm resumes, from the middle of the table to the end.

Everything is repeated as long as we can half the interval.

The need for the table to be ordered implies that its elements have a (key) component that belongs to a scalar type, and the search is done after this component [63].

For this search, we have made AGLOs which deals with the following aspects: definition of the notion, algorithm steps, operating principle, different use cases for different types of values, as well as the recognition of some implementations for different types of input data.

Question

Ce rol are algoritmul de cautare binara?

Answers

- Algoritmul de cautare binara este un algoritm de sortare folosit pentru a ascunde un element intr-un tablou neordonat. Algoritmul functioneaza pe baza tehnicii divide et impera.
 - Algoritmul de cautare binara este un algoritm de cautare folosit pentru a gasi un element intr-un tablou ordonat. Algoritmul functioneaza pe baza tehnicii divide et impera.
- Correct!

Evaluate

Feedbacks

Algoritmul de cautare binara este un algoritm de cautare folosit pentru a gasi un element intr-un tablou ordonat. Algoritmul functioneaza pe baza tehnicii divide et impera.

Fig.6.19. An implementation of the role of the binary search AGLO

The first AGLO targets the role of the binary search algorithm. The student must know that the binary search algorithm is a search algorithm used to find an element in an ordered table. The algorithm works based on the divide and rule technique. In this exercise, the false statement is the only one that will change during implementation.

In the next three exercises, the student is familiar with the situations in which it is specific to use the binary search algorithm, but also with those cases where it is not usable.

```
<scenario>
  <text>
Se genereaza trei numere aleatorii ca dimensiuni de tablou, si un tablou de
intregi, un tablou de caractere si un tablou de structuri.
  </text>
  <symbolname="nNoOfElements1" type="integer">random(5,15,0);</symbol>
  <symbolname="nNoOfElements2" type="integer">random(5,15,0);</symbol>
  <symbolname="nNoOfElements3" type="integer">random(5,15,0);</symbol>
  <symbolname="tab1" type="string">random_ordered_array({"nNoOfElements"
: v("nNoOfElements1"), "tabTypes":["integer"]}).toString()</symbol>
  <symbolname="tab2" type="string">random_ordered_array({"nNoOfElements"
: v("nNoOfElements2"), "tabTypes":["character"]}).toString()</symbol>
  <symbolname="tab3" type="string">random_ordered_array({"nNoOfElements"
: v("nNoOfElements3"), "tabTypes":["string","integer"]}).toString()</symbol>
</scenario>
```

Fig.6.20. The scenario section for situations in which binary search is applied AGLO

In Figure 6.20 the scenario section of the AGLO is represented, aiming at tables on which the binary search can be applied. In this AGLO, three unsigned integers and three tables are generated. In this exercise, the student receives three tables of different size: the first is made up of randomly generated integers, the second table consists of letters arranged in alphabetical order, generated randomly, and the other one is a table of structures, arranged by a key.

Question

In ce situatii practice este folosita cautarea binara?

Answers

- Pe sirul [31, 88, 149, 221, 238, 274, 339, 354, 434, 460, 499, 526, 588].
- Pe sirul ['b', 'd', 'f', 'g', 'h', 'j', 'k'].
- Pe sirul [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }] avand ca si cheie anul.
- Pe sirul [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }] avand ca si cheie numele.

Correct! Evaluate

Feedbacks

Cautarea binara nu se aplica pe siruri neordonate

Fig.6.21. Cases in which binary search is used

In the question section, the third table appears twice because, the first time the element is searched by the key according to which the elements are arranged, and the second time it is searched for another key according to which the elements are not ordered. We left the same table for the student to understand the need for the key according to which the table is ordered. The last variant that contains the key according to which the array is not ordered does not correspond to the application of the algorithm, so it is the wrong answer.

In the feedback part, the student receives a recapitulation of the notion; binary search cannot be applied to tables that are not ordered. In Figure 6.21 it is illustrated an implementation for AGLO targeting cases where the binary search is used.

In the next AGLO, the theme is the same but represented differently. Four numbers are randomly generated that will represent the dimensions of four tables. They are then generated: an array of natural numbers in ascending order, an array of integers ordered in descending order, an array of structures, and an array of integer numbers unordered.

Question

In ce situatii practice este folosita cautarea binara?

Answers

Pe sirul [33, 69, 95, 165, 211, 276, 353, 421, 456, 470, 504].

Pe sirul [72, 2, -52, -75, -95, -144, -218].

Pe sirul [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }] avand ca si cheie campul nume.

Pe sirul [225, 22, 176, 216, 168, 125, 86, 163, 138, 116, 26].

Correct! Evaluate

Feedbacks

Cautarea binara se aplica pe siruri ordonate crescator sau descrescator.

Fig.6.22. An implementation of cases in which binary search is used AGLO

The theme is the same but represented differently. Numbers are randomly generated that will represent the dimensions of four arrays, see Figure 6.22. They are then generated: an array of natural numbers in ascending order, an array of integers ordered in descending order, an array of structures, and an array of integer numbers unordered. In this situation, the idea that we have insisted is that it does not matter if the table is ordered in ascending or descending order.

In the next AGLO, the student receives three strings of integers and must choose on which of these tables binary search is not applicable. First, three unsigned integers are generated randomly, which will represent the dimensions of the three tables. Then, an ascending ordered table, a descending ordered table, and an

unordered table are randomly generated. The answer the student has to choose is the unordered table.

Question

In ce situatii practice nu se poate folosi cautarea binara?

Answers

Pe sirul [59, 115, 182, 217, 249, 302, 363, 432, 447].
 Pe sirul [78, 36, -14, -83, -103, -176, -242, -306, -326, -375].
 Pe sirul [284, 9, 110, 298, 292, 251, 7, 2, 271].
 Correct!

[Evaluate](#)

Feedbacks

Tabloul nu este ordonat nici crescator si nici descrescator.

Fig.6.23. An implementation of cases in which binary search can't be used AGLO

The following AGLO covers the steps of binary search. This is a very important thing that the student must master. To be able to understand how the algorithm works its steps are essential.

Theory

Cautarea binara se realizeaza pe tablouri ordonate. Daca se da tabloul: $tab=[7,15,20,34]$ si se cauta $x=20$, pasii algoritmului sunt urmatorii:
 $s=0$ $d=3$ $m=1$ $15 < 20$
 $s=2$ $d=3$ $m=2$ $20 = 20$

Question

Se da tabloul: $tab=[35,50,88,154,201,224,279,353,366]$ si se cauta $x=154$. Scrieti pasii algoritmului cautarii binare.

Fig.6.24. The theory and the question sections of an implementation of binary search steps AGLO

In Figure 6.24 the theory and questions sections are represented. In the theory section, the student is reminded that this search is applied only on ordered tables, then he is presented with a static example, a table with four whole elements, and a response model.

In the questions section, you receive the table on which you must apply the binary search algorithm.

Answers

```
s=0 d=8 m=4 154<201
s=0 d=3 m=1 154>50
s=2 d=3 m=3 154=154
```

You got 6.25 out of 10!

Evaluate

Fig.6.25. The answer section of an implementation of binary search steps AGLO

In Figure 6.25 we chose to give a partially correct answer to show how we chose to help the student in this case. The first two rows in the answer are correct, and the third is wrong. The correct answer would be two more lines. In this case, the student's grade is evaluated according to what percentage of the correct answer represents what he wrote.

Feedbacks

Cautarea binara se realizeaza prin tehnica divide et impera. La fiecare pas elementul cautat este comparat cu elementul din mijlocul tabloului. Dacă au valori egale, algoritmul se termină. Dacă elementul cautat e mai mare decât valoarea din mijlocul tabloului, algoritmul se reia, de la mijlocul tabloului până la sfârșit, iar dacă elementul cautat e mai mic decât valoarea din mijlocul tabloului, algoritmul se reia pentru elementele de la începutul tabloului până la mijloc.

```
s=0 s=0 OK
d=8 d=8 OK
m=4 m=4 OK
154<201 154<201 OK

s=0 s=0 OK
d=3 d=3 OK
m=1 m=1 OK
154>50 154>50 OK

s=2 s=2 OK
d=3 d=3 OK
m=3 m=2 NOT OK
154=154 154>88 NOT OK
```

Fig.6.26. The feedback section of an implementation of binary search steps AGLO

As you can see in Figure 6.26 in the feedback part the student first receives a notification with the theoretical part that should have been applied. After this, his answer is compared with the correct answer so that the student can see step by step what he did well and where the mistake occurred.

The last AGLO refers to the recognition of an implementation. The student receives an implementation of the binary search algorithm written in the C programming language.

```

<scenario>
<text>
Se schimba numele la variabile
- tablou: a b c d e f g h
- contor: i j k l m
- dimensiune: n o p q
- element de cautat: x y z t
</text>
<symbol name="tabTablou" type="array">["a","b","c","e","f","g","h"]
</symbol>
<symbol name="iTablou" type="integer">random(0,v("tabTablou").length-
1,0); </symbol>
<symbol name="tablou" type="string">v("tabTablou")[v("iTablou")];
</symbol>
<symbol name="tabContor" type="array">["i","j","k","l","m"]</symbol>
<symbol name="iContor" type="integer">random(0,v("tabContor").length-
1,0); </symbol>
<symbol name="contor" type="string">v("tabContor")[v("iContor")];
</symbol>
<symbol name="tabDimensiune" type="array">["n","o","p","q"]</symbol>
<symbol name="iDimensiune" type="integer">
random(0,v("tabDimensiune").length-1,0); </symbol>
<symbol name="dimensiune" type="string"> v("tabDimensiune")
[v("iDimensiune")]; </symbol>
<symbol name="tabElement" type="array">["x","y","z","t"]</symbol>
<symbol name="iElement" type="integer">
random(0,v("tabElement").length-1,0); </symbol>
<symbol name="element" type="string">v("tabElement")[v("iElement")];
</symbol>
</scenario>

```

Fig.6.27. The scenario section for the binary search recognition AGLO

As can be seen in Figure 6.27, although the exercise does not seem very complicated for its implementation, many variables are needed. The beginning of the scenario describes how the variables are instantiated:

- the table variable can have one of the values: *a, b, c, d, e, f, g* or *h*;
- the meter variable can have one of the values: *i, j, k, l* or *m*;
- the dimension variable can have one of the values: *n, o, p* or *q*;
- the variable element to be searched for can have one of the values: *x, y, z,*

or *t*.

This choice gives us the fact that at each instant the student will receive a different version so that he can recognize the elements that define this algorithm.

In response, the student must choose one of the search variants: linear, binary, or by interpolation.

It must know that it is a binary search because we have specific left and right indexes within the range that this algorithm applies. It is also important to know that it is not a linear search because it is not sequentially searched for, item by item. There is no interpolation search because we do not have the pivot interpolation formula.

Ce algoritm reprezinta secventa de cod de mai jos ?

```
s=0;
d=0-1;
do
{
    i=(s+d)/2;
    if(t>c[i])
        s=i+1;
    else
        d=i-1;
}
while((c[i]!=x) && (s<=d));
```

Answers

1. Cautare liniara
2. Cautare binara
3. Cautare prin interpolare

Evaluate

Correct!

Feedbacks

Nu este vorba de cautare binara deoarece nu se cauta secvential element cu element.
 Este vorba de o cautare binara pentru ca avem indecsii specifici pentru stanga si dreapta intervalului pe care acest algoritm se aplica.
 Nu este vorba de o cautare prin interpolare pentru ca nu avem formula de interpolare a pivotului.

Fig.6.28. An implementation of binary search recognition AGLO

In Figure 6.28 illustrates an implementation of AGLO aimed at recognizing the binary search algorithm. In the feedback section, the student is reminded of the significant ideas regarding the targeted notions.

6.1.4. Search by interpolation AGLOs

From a theoretical point of view, we consider that we have a table of integers called $tab[]$ of length N .

The search interpolation method is efficient if:

- the number of elements N is very large and
- the values of the elements of the table have a uniform distribution in the range of $tab[1], \dots, tab[N]$
- the search element, x , be within the range of $tab[1], \dots, tab[N]$.

If the search element is not in the range of $tab[1], \dots, tab[N]$ there is a risk that the calculated value of m will exceed N .

This search is similar to binary search but uses another formula for calculating m . The formula for calculating m , is $m = s + (x - \text{tab}[s]) * (d - s) \text{ div } (\text{tab}[d] - \text{tab}[s])$. Here s represents the index of the first element and d the index of the last element in the table on which the search is applied.

For this search, we have made AGLOs which deals with the following aspects: definition of the notion, algorithm steps, operating principle, different use cases for different types of values, as well as the recognition of some implementations for different types of input data.

The first AGLO refers to the understanding of the notion.

Question

Ce rol are algoritmul de cautare prin interpolare?

Answers

- Algoritmul de cautare binara este un algoritm de sortare folosit pentru a ascunde un element intr-un tablou ordonat. Algoritmul funcționează pe baza tehnicii divide et impera si foloseste o formula de interpolare a pivotului.
- Algoritmul de cautare binara este un algoritm de cautare folosit pentru a găsi un element intr-un tablou ordonat. Algoritmul funcționează pe baza tehnicii divide et impera si foloseste o formula de interpolare a pivotului.

Correct!

[Evaluate](#)

Feedbacks

Algoritmul de cautare binara este un algoritm de cautare folosit pentru a găsi un element intr-un tablou ordonat. Algoritmul funcționează pe baza tehnicii divide et impera si foloseste o formula de interpolare a pivotului.

Fig.6.29. An implementation of search by interpolation AGLO

In this exercise, the student must choose the true sentence from the two variants he receives, as can be seen in Figure 6.29.

The true sentence in which the notion is presented does not change in different implementations, while the false sentence has several variants from which a random one is chosen.

The following AGLO addresses practical situations in which interpolation search is used.

The theory sections are the same for those AGLOs, because in solving them the same theoretical part is necessary. In these sections, it is recalled that this method is efficient if n is very large and the values of the elements of the array have a uniform distribution in the range $\text{tab}[1], \dots, \text{tab}[n]$. Applying the search by interpolation requires the search element to be inside the interval $\text{tab}[1], \dots, \text{tab}[n]$.

The student receives four variants regarding specific situations of interpolation search. Depending on the variants it receives, what is true will be the fact that this search is applied on large, ordered strings, with the elements evenly distributed in the range of the first and last element, and the element to be searched must be in this range.

What will be a false variant will refer to different implementations on unordered tables or tables of structures, see Figure 6.30.

Theory

Algoritmul de căutare prin interpolare este un algoritm de căutare folosit pentru a găsi un element într-o listă ordonată.

Question

In ce situatii practice este folosita cautarea prin interpolare?

Answers

Cand dimensiunea n este foarte mare și valorile elementelor tabloului au o distributie uniformă în intervalul $a[1], \dots, a[n]$.

Cand elementul de căutat, x, se afla în interiorul intervalului $a[1], \dots, a[n]$.

Cand tabloul este neordonat.

Pe sirul `{{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }}` avand ca si cheie anul.

Correct!

[Evaluează](#)

Feedbacks

Cautarea prin interpolare se aplica pe siruri ordonate.

Fig.6.30. An implementation of practical situations in which interpolation search is used AGLO

Theory

Această metodă este eficientă în cazul în care n este foarte mare și valorile elementelor tabloului au o distributie uniformă în intervalul $tab[1], \dots, tab[n]$. Aplicarea căutării prin interpolare necesită ca elementul de căutat, x, să se afle în interiorul intervalului $tab[1], \dots, tab[n]$, altfel apare riscul ca valoarea calculată a lui m să depășească valoarea lui n.

Question

In ce situatii practice este folosita cautarea prin interpolare?

Answers

Pe sirul 41,61,81,101,121,141,161,181,201,221,241,261,281,301,321,341,361,381,401,421,441,461,481.

Pe sirul [10, 5, 0, -5, -10, -15, -20, -25, -30, -35, -40, -45, -50, -55, -60, -65, -70, -75, -80, -85, -90, -95, -100].

Pe sirul `{{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"volvo", an:2007, }, {nume:"mercedes", an:2015, }}`.

Pe sirul [257, 126, 158, 186, 179, 189, 190, 250, 238, 34, 5, 129, 151, 283, 20, 242, 164, 222, 58, 60, 32, 91, 55].

Correct!

[Evaluează](#)

Feedbacks

Cautarea prin interpolare se aplica pe siruri ordonate crescator sau descrescator.

Fig.6.31. An implementation of types of data to which interpolation sorting is applied AGLO

The next AGLO refers to strings to which this method can be applied.

As can be seen in Figure 6.31, the student receives several variants of strings from which he must choose the ones on which the application is suitable by interpolation.

In the scenario section, the following are declared and initialized for strings:

- a table of integers in ascending order,
- a table of integers, in descending order,
- a table of structures,
- a table of random integers (unordered).

To simulate a larger number of elements, specific to this method, we chose that the lengths of the tables should be numbers greater than 15. In this exercise, the correct variants are the tables of integers ordered ascending or descending.

The next AGLO refers to the determination of strings on which interpolation search cannot be used.

Theory

Această metodă este eficientă în cazul în care n este foarte mare și valorile elementelor tabloului au o distribuție uniformă în intervalul $a[1], \dots, a[n]$. Aplicarea căutării prin interpolare necesită ca elementul de căutat, x , să se afle în interiorul intervalului $a[1], \dots, a[n]$, altfel apare riscul ca valoarea calculată a lui m să depășească valoarea lui n .

Question

În ce situații practice nu se poate folosi căutarea prin interpolare?

Answers

Pe sirul [31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 201, 211, 221, 231, 241, 251, 261].
 Pe sirul [34, 18, 2, -14, -30, -46, -62, -78, -94, -110, -126, -142, -158, -174, -190, -206, -222, -238, -254, -270, -286, -302, -318, -334].
 Pe sirul [152, 58, 146, 24, 6, 172, 197, 294, 91, 77, 196, 62, 3, 192, 88, 149, 136, 178, 57, 277, 97, 112, 29, 151].

Correct! Evaluate

Feedbacks

Tabloul nu este ordonat nici crescator si nici descrescator.

Fig.6.32. An implementation of strings on which interpolation search cannot be used AGLO

The next AGLO, see Figure 5.32 targets cases in which interpolation search cannot be applied. The student receives three variants of tables from which he must choose the improper ones.

In the scenario section, the following are declared and initialized for strings:

- a table of integers in ascending order,

- a table of integers, in descending order,
- a table of random integers (unordered).

It is obvious to those who know the notion that the only option to check is the sequence of unordered integers. In the feedback section, the student receives confirmation of this fact.

The first AGLOs presented are not very sophisticated, but they aim at the correct implementation of the notion. The student must know the specific characteristics of the notion to be able to apply it correctly. You also need to know the practicalities to know when it is possible to use this method.

The next AGLO aims to know the steps of the search by interpolation.

In the theory section, the student receives the formula for the pivot m and a static example. The example also has the role of showing the student under what form the answer should be written.

In this exercise, a table of 10 to 15 elements is randomly generated. To respect the real conditions of this search, we evenly distributed the elements between the first and the last element. Also, the element to be searched for is randomly generated in the interval between the first and the last element of the table.

Theory

Cautarea prin interpolare se realizeaza pe tablouri ordonate. Se asemana cu cautarea binara, doar ca are o formula de interpolare pivotului $m = s + (x - \text{tab}[s]) * ((d - s) / (\text{tab}[d] - \text{tab}[s]))$.

Pentru un tablou $\text{tab} = [19, 35, 51, 67, 83, 99, 115, 131, 147, 163]$ si se cauta $x = 67$ pasii algoritmului sunt urmatoarii:
 $s = 0$ $d = 9$ $m = 3$ $67 = 67$
 DA, elementul a fost gasit

Question

Daca se da tabloul: $\text{tab} = [23, 35, 47, 59, 71, 83, 95, 107, 119, 131, 143, 155]$ si se cauta $x = 59$ pasii algoritmului sunt urmatoarii:

Answers

$s = 0$ $d = 11$ $m = 3$ $59 = 59$
 DA, elementul a fost gasit

[Evaluate](#)

You got 10 out of 10!

Feedbacks

Cautarea prin interpolare se realizeaza asemenea cautarii binare.

$s = 0$ $s = 0$ OK
 $d = 11$ $d = 11$ OK
 $m = 3$ $m = 3$ OK
 $59 = 59$ $59 = 59$ OK

Fig.6.33. An implementation interpolation search steps AGLO

An implementation of AGLO targeting the search by interpolation steps is presented in Figure 6.32. And here it can be seen that the student's answer is analyzed step by step in case of a mistake so that he can easily identify it, and the grade obtained should be partial.

The following AGLO aims to recognize the implementation of the search algorithm by interpolation.

For this, in the question section is represented as a program sequence written in the C programming language. This sequence represents the search by interpolation. Within each implementation the names of the variables in this sequence change.

In the answer section, the student must choose one of the three options he receives: linear search, binary search, and interpolation search.

Ce algoritm reprezinta secventa de cod de mai jos ?

```
s=0;
d=0-1;
while (s <= d && x >= b[s] && x <= b[d])
{
    m=s+(x-b[s])*((d-s)/(b[d]-b[s]));
    if(x>b[m])
    {
        s=m+1;
    }
    else
    {
        d=m-1;
    }
}

if(b[m]==x)
    printf("coincidenta la pozitia %d",m);
else
    printf("elementul cautat nu exista in tablou");
```

Answers

1. Cautare liniara
 2. Cautare binara
 3. Cautare prin interpolare
 Correct!

Feedbacks

Nu este vorba de cautare binara deoarece nu se cauta secvential element cu element.
 Nu este vorba de o cautare binara pentru ca nu se calculeaza mijlocul intervalului.
 Este vorba de o cautare prin interpolare pentru ca avem formula de interpolare a pivotului.

Fig.6.34. An implementation of recognition of the interpolation search AGLO

As shown in Figure 6.34, in the feedback section are recalled the characteristics of the three types of searches that appear between the answer options. So, the following are presented:

- it is not about binary search because it is not searched sequentially item by item.

-it is not a binary search because the middle of the interval is not calculated.

-it is an interpolation search because we have the pivot interpolation formula.

6.1.5. Insertion sort AGLOs

For the insertion sorting, the basic idea is to insert a certain element in the already sorted table of its predecessors [64].

For this search, we have made AGLOs which deals with the following aspects: definition of the notion, algorithm steps, operating principle, different use cases for different types of values, as well as the recognition of some implementations for different types of input data.

Theory

Algoritmul de sortare prin insertie insereaza in zona de tablou sortata cate un singur element.

Question

Ce rol are algoritmul de sortare prin insertie?

Answers

- 1.Sa insereze cate 4 elemente la fiecare pas si sa le sorteze doar pe ele
- 2.Sa insereze cate 1 element la fiecare pas si sa le sorteze pe toate din tablou.
- 3.sa insereze cate 6 elemente la fiecare pas si sa le sorteze pe cele existente initial in tablou

Correct! Evaluate

Feedbacks

Algoritmul de sortare prin insertie insereaza in zona de tablou sortata cate un singur element. Evident ca toate elementele din tablou vor fi la final sortate.

Fig.6.35. An implementation of insertion sorting definition AGLO

In the first AGLO, we give the student three sentences, two false and one true, see Figure 6.35. The correct sentence is static, and the two false sentences are different at each implementation.

In the feedback section, we remind the student what is the role of the insertion-sorting algorithm.

The next AGLO aims to help the student to identify practical situations of using the insertion-sorting algorithm.

In the question section are presented two options:

- a static one does not change: *When ordering is needed, simple to implement, to an unordered table;*

- a dynamic sentence that contains elements that change their value.

Both statements are true at every implementation.

The specific features of sorting inserts are mentioned in the feedback section.

The student must remember that this sorting applies to tables with a small number of elements.

The screenshot displays a user interface for an AGLO (Assessment Goal Learning Object) with four distinct sections:

- Theory:** A green header bar containing the text "Sortarea prin insertie se aplica la tablouri cu numar redus de elemente."
- Question:** A red header bar containing the question "In ce situatii practice este folosit algoritmul de sortare prin insertie?"
- Answers:** A blue header bar containing two checked options: "1.Cand tabloul ce trebuie ordonat are doar 27 elemente." and "2.Cand e nevoie de o ordonare, simplu de implementat, la un tablou neordonat." Below the options, it says "Correct!" and includes an "Evaluate" button.
- Feedbacks:** A yellow header bar containing two feedback items: "1.Sortarea prin insertie se aplica la tablouri cu numar redus de elemente." and "2.Sortarea prin insertie se aplica pe tablouri neordonate de mici dimensiuni."

Fig.6.36. An implementation of practical situations to which insertion sorting applies AGLO

The following AGLO identifies several types of data collections to which insertion sorting can be applied.

In the scenario section, see Figure 6.36, four integers are randomly generated that will represent the number of elements of some tables. We chose each interval in which these numbers are generated to be different. The following are then generated:

- a table of integers, which represents the primitive types;
- a table of strings, which represents any types of elements;
- a table of structures with a string field and an integer field, representing a non-scaling data collection;
- a table of structures with an address field and a string field, representing a scaling data collection.

```

<scenario>
  . . .
  <symbol name="nNoOfElement1" type="integer">random(7,12,0);</symbol>
  <symbol name="nNoOfElement2" type="integer">random(5,12,0);</symbol>
  <symbol name="nNoOfElement3" type="integer">random(5,10,0);</symbol>
  <symbol name="nNoOfElement4" type="integer">random(5,7,0);</symbol>
  <symbol name="tab1" type="string"> random_array({"nNoOfElements" :
v("nNoOfElement1"), "tabTypes" : ["integer"]}).toString(); </symbol>
  <symbol name="tab2" type="string">random_array({"nNoOfElements" :
v("nNoOfElement2"), "tabTypes" : ["string"]}).toString();</symbol>
  <symbol name="tab3" type="string"> random_array({"nNoOfElements" :
v("nNoOfElement3"), "tabTypes" : ["string","integer"]}).toString();</symbol>
  <symbol name="tab5" type="string"> random_array({"nNoOfElements" :
v("nNoOfElement4"), "tabTypes" : ["address","string"]}).toString();</symbol>
</scenario>

```

Fig.6.37. The scenario section of types of data collections on which insertion sorting applies AGLO

Question

Pe de ce fel de tipuri de colectii de date se poate aplica sortarea prin insertie?

Answers

pe tablou de siruri de caractere cu continut de diferite tipuri ["audi", "bmw", "mercedes", "toyota", "lexus", "honda", "acura"];

pe colectie de elemente de diverse tipuri [{name:"audi", an:1995, }, {name:"bmw", an:2001, }, {name:"logan", an:2016, }, {name:"mercedes", an:2015, }, {name:"opel", an:2011, }, {name:"volvo", an:2007, }];

pe tablou de tipuri primitive [186, 164, 191, 145, 141, 200, 123, 50, 125];

pe tablou de structuri [{adresa:0x26, nume:"alin", }, {adresa:0x33, nume:"barbu", }, {adresa:0x42, nume:"cezar", }, {adresa:0x53, nume:"dorin", }];

Correct!

[Evaluate](#)

Feedbacks

Da, putem aplica algoritmul de sortare prin insertie pe tablouri cu elemente de tip string chiar daca continutul lor este de diverse tipuri.

Da, putem aplica algoritmul de sortare prin insertie pe tablouri cu elemente de tip primitiv.

Da, putem aplica algoritmul de sortare prin insertie pe tablouri de structuri sortarea facandu-se doar dupa un singur camp la un moment dat.

Fig.6.38. An implementation of types of data collections on which insertion sorting applies AGLO

Of the options presented in each implementation, only one is not correct, see Figure 6.38.

In the feedback section, the student receives a recapitulation of the notions that are applied for the accomplishment of this exercise. If he chooses the wrong option which is not represented in the figure above, the student is reminded that he cannot apply the inserting sort algorithm on collections of elements of various types that do not have an ordering rule.

The following AGLO aims at recognizing the implementation of the inserting search algorithm in the C programming language. Three code variants are implemented, one of which is chosen randomly at each implementation.

The student receives three answers options: insertion, selection, and interchange. The correct option is the insertion.

Question

Se da algoritmul urmatore reprezentand o sortare.

```
int v[100];
int k,j;
int x;
for(k=1;k<n;k++)
{
    x=v[k];
    j=k-1;
    while(v[j]>x && j>=0)
    {
        v[j+1]=v[j];
        j=j-1;
    }
    v[j+1]=x;
}
```

El reprezinta o sortare prin:

Answers

insertie
 selectie
 interschimbare

Correct!

Fig.6.39. An implementation of insertion sorting recognition AGLO

An implementation of the AGLO that aims the recognition of an implementation of the insertion-sorting algorithm in the C programming language is presented in Figure 6.39.

Since an algorithm can have several variants, we chose to make two AGLOs aimed at recognizing an implementation.

In the second AGLO, the student receives a program that uses a sorting algorithm. The question is whether the algorithm used in that program is insert sorting. In this case, the student has to choose only between the Yes or No options.

At each implementation, a number in the range 10 - 15 and a table of integers of this size are randomly generated.

The student must observe the two cycling instructions for traversing and for inserting the elements, respectively.

The following AGLO is very important because it aims to know the steps of the insertion sorting algorithm.

Theory

Algoritmul de sortare prin inserție construiește pas cu pas lista de elemente sortate, adăugând la ea câte un element la un moment dat. La fiecare pas un element este extras din lista inițială și este introdus în lista de elemente sortate. Elementul este inserat în poziția corectă în lista sortată, astfel încât ea să rămână sortată în continuare.

Care sunt versiunile intermediare ale sirului `tab=[57,35,26,47,28,38]`, în cadrul sortării prin inserție?

```
57 35 26 47 28 38
35 57 26 47 28 38
26 35 57 47 28 38
26 35 47 57 28 38
26 28 35 47 57 38
26 28 35 38 47 57
```

Question

Care sunt versiunile intermediare ale sirului `tab=[115,85,88,160,179,298]`, în cadrul sortării prin inserție?

Answers

```
115 85 88 160 179 298
85 115 88 160 179 298
85 88 115 160 179 298
85 88 115 160 179 298
85 88 115 160 179 298
85 88 115 160 179 298
```

Evaluate

You got 10 out of 10!

Feedbacks

```
115 115 OK
85 85 OK
88 88 OK
160 160 OK
179 179 OK
298 298 OK
OK
85 85 OK
115 115 OK
```

Fig.6.40. An implementation of insertion sorting steps AGLO

In the theory section, it is presented in brief how the algorithm works. The insert sort algorithm builds gradually the list of sorted items, adding one item at a time. At each step, an item is extracted from the initial list and entered in the sorted list. The item is inserted in the correct position in the sorted list so that it remains sorted. A static example is also presented so that the student knows in what form the answer should be represented.

To formulate the question, a number between 5 and 8 and a table of this size are randomly generated.

An implementation of the AGLO which aims steps of the insertion sorting algorithm is presented in Figure 6.29.

In the feedback section, the student's response is compared item by item with the correct answer. In this way, in case of a wrong answer, the student can notice a mistake and thus can correct it.

Also, the answer is evaluated proportionally with the correctly written steps.

6.1.6. Selection sort AGLOs

The algorithm checks each number and starts with the smallest one. The next number after $tab[0]$ is the smallest of the remaining numbers but greater than the $tab[0]$. The step is repeated until the table is sorted.

For this search, we have made AGLOs which deals with the following aspects: definition of the notion, algorithm steps, operating principle, different use cases for different types of values, as well as the recognition of some implementations for different types of input data.



Theory

Sortarea prin selectie se aplica la tablouri cu numar redus de elemente.

Question

In ce situatii practice este folosit algoritmul de sortare prin selectie?

Answers

1. Cand tabloul ce trebuie ordonat are doar 12 elemente.

2. Cand e nevoie de o ordonare, simplu de implementat, la un tablou neordonat.

Correct! [Evalueate](#)

Feedbacks

1. Sortarea prin selectie se aplica pe tablouri de cateva elemente.

2. Sortarea prin selectie se aplica pe tablouri neordonate de mici dimensiuni.

Fig.6.41. An implementation of selection sorting algorithm use situations AGLO

In the first AGLO, the student is taught to identify the situations of using the selection sorting algorithm.

The main idea of the exercise is that selection sorting is applied to small unordered tables.

The two answer options are true for each implementation. One of the sentences changes with each implementation and the other does not.

Figure 6.30 shows an implementation of AGLO that targets the practical situations in which the selection sorting algorithm is used.

The following AGLO aims for the student to know on which tables the algorithm can be applied.

Six integers are randomly generated in the range 5 to 15 that will represent the dimensions of the six tables to be initialized. The following are generated:

- a table of unordered integers,
- a table of randomly selected characters,
- a table of randomly selected strings,
- a table of integers in ascending order,
- a table of characters in alphabetical order,
- a table of strings in alphabetical order.

All generated variants are true. The purpose of the exercise is to present to the student the fact that the selection sorting algorithm can be applied on any table with elements of any type as long as there is the possibility to compare two elements of it. Applying the algorithm to an ascending table makes sense only if you want to sort it in descending order or vice versa, otherwise, the table remains unchanged after applying the sorting algorithm.

The next AGLO targets data collections on which the selection sorting algorithm is used.

In this case, four tables of different sizes are randomly generated:

- a table of strings with different types of content,
- a table of structures with an address type field and a string type field, this represents a collection of elements of various types,
- a table of unordered integers, which represents an array of primitive types,
- an array of structures that can be ordered according to one of the fields.

Among these variants, the table that represents a collection of elements of various types is the one that does not meet the necessary conditions to apply the selection sorting algorithm.


Question

Algoritmul de sortare prin selectie se poate aplica pe:

Answers

- sirul de intregi [282, 80, 122, 52, 248, 208, 236, 210, 138, 62];
- sirul aleator de caractere ['a', 'b', 'c', 'd', 'f', 'h', 'j', 'm', 'p', 's'];
- sirul aleator de siruri de caractere ["audi", "bmw", "mercedes", "toyota", "lexus", "honda", "acura"];
- sirul ordonat de intregi [22, 64, 112, 142, 195, 265, 340, 409, 452, 520];
- sirul ordonat de caractere ['b', 'd', 'f', 'g', 'h', 'j', 'l', 'n', 'p', 'r'];
- sirul ordonat de stringuri ["audi", "bmw", "lexus", "mercedes"];

Wrong!

 Evaluate

Feedbacks

Algoritmul de sortare prin selectie se poate aplica pe orice tablou cu elemente de orice tip atata timp cat exista posibilitatea de a compara doua elemente ale acestuia. Algoritmul de sortare prin selectie are sens sa fie aplicat pe un tablou neordonat de elemente. Aplicarea sa pe un tablou ordonat crescator are sens doar daca se doreste sortarea sa descrescatoare sau viceversa. Altfel tabloul ramane nemodificat in urma aplicarii algoritmului de sortare.

Fig.6.42. An implementation of data types on which the selection-sorting algorithm can be applied AGLO

Question

Pe de ce fel de tipuri de colectii de date se poate aplica sortarea prin selectie?

Answers

- pe tablou de siruri de caractere cu continut de diferite tipuri ["audi", "bmw", "mercedes", "toyota", "lexus", "honda", "acura"];
- pe colectie de elemente de diverse tipuri [{adresa:0x26, nume:"alin", }, {adresa:0x33, nume:"barbu", }, {adresa:0x42, nume:"cezar", }, {adresa:0x53, nume:"dorin", }];
- pe tablou de tipuri primitive [182, 204, 211, 93, 25, 14, 188, 33, 138, 235, 110];
- pe tablou de structuri [{nume:"audi", an:1995, }, {nume:"bmw", an:2001, }, {nume:"logan", an:2016, }, {nume:"mercedes", an:2015, }, {nume:"opel", an:2011, }, {nume:"volvo", an:2007, }];

Correct!

[Evaluate](#)

Fig.6.43. An implementation of data collections on which the selection sorting algorithm can be applied AGLO

Figure 6.43 shows an implementation of AGLO that targets the collections of different types of data on which the selection search algorithm can be applied.

Theory

Algoritmul de sortare prin selectie construieste pas cu pas lista de elemente sortate, adaugand la ea cate un element la un moment dat. La fiecare pas extras cel mai mic element din lista initiala si este introdus in lista de elemente sortate. Elementul este inserat la sfarsit in lista sortata, astfel incat ea sa ramana sortata in continuare.

Exemplu:

```
Care sunt versiunile intermediare ale sirului tab=[19,44,59,33,50,30], in cadrul sortarii prin selectie?
Raspuns:
19 44 59 33 50 30
19 30 59 33 50 44
19 30 33 59 50 44
19 30 33 44 50 59
19 30 33 44 50 59
```

Question

Care sunt versiunile intermediare ale sirului tab=[166,227,155,154,205], in cadrul sortarii prin selectie?

Answers

```
154 166 227 155 205
154 155 166 227 205
154 155 166 227 205
154 155 166 205 227
```

[Evaluate](#)

You got 6.52 out of 10!

Feedbacks

```
154 154 OK
166 227 NOT OK
227 155 NOT OK
155 166 NOT OK
205 205 OK

154 154 OK
155 155 OK
166 227 NOT OK
227 166 NOT OK
205 205 OK

154 154 OK
155 155 OK
166 166 OK
227 227 OK
205 205 OK

154 154 OK
155 155 OK
166 166 OK
205 205 OK
227 227 OK
```

Fig.6.44. An implementation of the Selection Sorting Algorithm Steps AGLO

The next AGLO aims the steps of the selection sort algorithm, that is, the intermediate versions of a table to which the selection search is applied.

In the theory section, the student receives a brief description of the algorithm. The selection sort algorithm builds step by step the list of sorted items, adding one item at a time. At each step, the smallest item from the initial list is extracted and entered into the sorted list. The item is finally inserted in the sorted list so that it remains sorted. The section also includes a static example through which the student can practically understand the steps and also have it as a model for the form of the answer to be given.

To complete the question, an integer, n , is first generated in the range 5 - 8. Then the n elements of the table are randomly generated.

The answer is represented by the intermediate versions of the table that are obtained at each activity loop of the algorithm.

In the feedback section, the student's answer is compared item by item with the correct answer. Thus, if it is wrong, as can be seen in Figure 6.44, it is possible to identify exactly which is the step where the error occurred.

Figure 6.44 shows an implementation of AGLO that targets the steps of the selection sorting algorithm with a partially correct answer.

6.1.7. Bubble sort AGLOs

By this method, the table is scanned and each element is compared with its successor. If they are not in order, the two elements are interchanged. The table is traversed several times, until at a complete journey no more interchange between the elements is performed, in this case, the table is sort [65].

The first AGLO aims the mode of operation of the algorithm, i.e. the intermediate versions of the table on which the algorithm is applied.

```
<scenario>
. . .
<symbol name="n" type="integer">random(5,7,0);</symbol>
<symbol name="tab" type="array">random_array({"nNoOfElements" :
v("n"), "tabTypes":["int"]}).tabElements;</symbol>
<symbol name="tab1" type="string">v("tab").toString();</symbol>
<symbol name="is" type="object">bubble_sort({"array":v("tab")});
</symbol>
<symbol name="i2" type="string">v("is").getSteps();</symbol>
</scenario>
```

Fig.6.45. The scenario section of the bubble-sorting algorithm steps AGLO

In the scenario section, a number n is randomly generated in the range 5 to 7, this represents the size of the table. We have chosen a smaller number of elements because here we are primarily looking at understanding the algorithm steps, and this requires accessible cases. A table of n elements is then randomly generated. Based on this table, a *bubble sort* object is initialized, which has the method to determine the steps of the bubble sorting algorithm.

Theory

Algoritmul de sortare prin interschimbare, pentru sirul `tab=[1,10,7,3,4]` are urmatoarele versiuni intermediare:
 1 3 10 7 4
 1 3 4 10 7
 1 3 4 7 10
 1 3 4 7 10

Question

Care sunt versiunile intermediare ale sirului `tab=[179,242,156,31,293]`, in cadrul sortarii prin interschimbare(bubble sort)?

Answers

```
31 179 242 156 293
31 156 179 242 293
31 156 179 242 293
31 156 179 242 293
```

[Evaluate](#)

You got 10 out of 10!

Feedbacks

```
31 31 OK
179 179 OK
242 242 OK
156 156 OK
293 293 OK
OK

31 31 OK
156 156 OK
179 179 OK
242 242 OK
293 293 OK
OK

31 31 OK
```

Fig.6.46. An implementation of the bubble-sorting algorithm steps AGLO

In Figure 6.46 it can be seen that in the theory section is presented a static example that has the role of showing the student what are the steps of the algorithm and in what form the answer must be written correctly.

In the feedback section, the student's answer is compared to the correct answer. As in the other AGLOs that aim at the steps of an algorithm, and in this case, we are interested in the student to see, if he gave a partial or incorrect answer, where exactly he was wrong.

In the next AGLO, the aim is for the student to know how many interchanges are made at a crossing of a table in the bubble sort algorithm.

In the theory section, the student is presented with a static example, a table of integers and the answer to the question: How many changes are made at the first crossing of that table.

In the question section, a table of integers is randomly generated, with 5 to 10 elements.

The answer is represented by a number, the number of interchanges that are made at the first traversal of the table within the sorting algorithm.

Theory

Cate interschimbari se fac la prima parcurgere a sirului tab=[1,10,7,3,4] in cadrul sortarii prin interschimbare(bubble sort)?
3

Question

Cate interschimbari se fac la prima parcurgere a sirului tab=[285,32,7,157,211,37,240], in cadrul sortarii prin interschimbare(bubble sort)?

Fig.6.47. An implementation of the bubble-sorting algorithm interchanges AGLO

Figure 6.47 shows an implementation of the bubble sorting algorithm interchanges AGLO.

6.1.8. Shell sort AGLOs

Shell sort is a highly efficient sorting algorithm and is quite used for medium-sized data sets.

Shell sort can be seen as a generalization of the bubble sorting algorithm. The method begins by sorting the pairs of elements apart from each other, progressively reducing the gap (h) between the elements to be compared. Starting with distant elements, it can move some of the elements out of place faster than a simple change of neighbor. This gap is calculated based on Knuth's formula as $h=h*3+1$, where h has the initial value 1 [65].

The first AGLO aims to determine the series of a table of integers for a random h between 3 and 5. To be as convenient as possible the generated example we chose to generate the length of the string between 10 and 15.

In the theory section, the student has presented as an example the series obtained for a table of 15 elements and $h = 4$. This model also shows the student how to write the answer he will give.

In the feedback section, the student's answer is compared sequentially with the correct answer. In this way, we aim again that in case of a wrong or partial answer the student can see where the mistake occurred.

Theory

Seriile sirului `tab=[25,38,60,46,18,40,9,45,36,56,66,17,7,16,35]` pentru `h=4` sunt urmatoarele:
 25 18 36 7
 38 40 56 16
 60 9 66 35
 46 45 17

Question

Daca se da tabloul: `tab=[163,119,55,137,122,204,245,244,6,75,17,257,15]` si `h=4` Scrieti pe cate o linie seriile care vor fi sortate independent.

Answers

```
163 122 6 15
119 204 75
55 245 17
137 244 257
```

[Evaluate](#)

You got 10 out of 10!

Feedbacks

```
163 163 OK
122 122 OK
6 6 OK
15 15 OK
OK
119 119 OK
204 204 OK
75 75 OK
OK
55 55 OK
```

Fig.6.48. An implementation of the Series in Shell Sorting Algorithm AGLO

Figure 6.48 shows an implementation of the AGLO which aims the way it is formed series in the shell sorting algorithm. The example is given with the correct answer.

The following AGLO aims to determine all states of the first series for a random `h`.

The theory section presents an example applied to a series of 15 elements and `h = 4`. The answer is written in the form that the answer that the student will have to write must-have.

In the question section, you will find a table of integers of length 10-15 generated randomly and an h in the range 3-5. The student must write on one line at a time the sequence of states for the first series generated on the case he receives, which will be sorted independently.

Theory

Pentru sirul `tab=[25,38,60,46,18,40,9,45,36,56,66,17,7,16,35]` cu $h=4$ starile primei serii sunt:

```
25 18 36 7
18 25 7 36
18 7 25 36
7 18 25 36
7 18 25 36
```

Question

Daca se da tabloul: `tab=[60,252,188,241,30,239,156,74,172,10,291,195]` si $h=3$ Scrieti pe cate o linie succesiunea starilor pentru prima serie care va fi sortata independent.

Answers

```
60 241 156 10
60 156 10 241
60 10 156 241
10 60 156 241
```

[Evaluate](#)

You got 10 out of 10!

Feedbacks

```
60 60 OK
241 241 OK
156 156 OK
10 10 OK
OK
```

Fig.6.49. An Implementation of the Sequence of States for the First Series in Shell Sorting AGLO

Because it is a succession of steps and in this case, in the feedback section the student's answer is compared sequentially with the correct answer. Figure 6.37 shows an implementation of AGLO that aims at the sequence of states for the first series that will be sorted by shell sort.

6.1.9. Quicksort AGLOs

Quick sort is a famous sorting algorithm, which in practice is faster than other sorting algorithms because its inner loop has efficient implementations on most architecture.

Quicksort performs sorting based on a *divide et impera* strategy. Thus, it divides the sorting list into two sublists that are easier to sort [64]. The steps of the algorithm are:

- Choose an item from the list, called a pivot
- Rearrange the list so that all items smaller than the pivot are placed before the pivot and all items larger than after the pivot. After this partitioning, the pivot is in its final position.
- Recursively sorts the sublist of items smaller than the pivot and the sublist of items larger than the pivot.

A list of size 0 or 1 is considered sorted.

The first AGLO targets the index and value of the pivot element for a quick sort algorithm.

Theory

Pivotul este ales ca fiind elementul din mijloc al sirului.

Question

Care este indicele si valoarea elementului pivot al sirul tab=[126,51,67,239,175,30], pentru sortarea Quick Sort?

Answers

You got 10 out of 10!

Fig.6.50. An implementation of the index and value of the pivot element for quick sort AGLO

The student receives the theoretical notion that must be applied in the exercise, namely that the pivot is the middle element of the table, see Figure 6.50.

The student must apply this to a randomly generated integer table with some elements in the range 6 - 10.

The expected answer consists of two numbers, the first represents the index of the element, and the second the value of the element.

The next AGLO aims for the student to know how to identify the first element bigger than the pivot. Also, if it does not exist, the student must know that the answer will represent the value of the pivot.

In the theory section, is presented a static example containing a table of seven integer elements.

Next, the student receives a table of randomly generated elements with a length between 5 and 8 and in which he must identify the first element bigger than the pivot. Although the notion is quite easy, it is very important in understanding the steps of the quick sorting algorithm.

The screenshot displays a user interface for a learning activity. It is divided into several colored sections: a green 'Theory' section, a red 'Question' section, a blue 'Answers' section, and a yellow 'Feedbacks' section. The 'Theory' section contains text about an array and a pivot. The 'Question' section asks the user to find the first element greater than a pivot in a specific array. The 'Answers' section has a text input field containing '210' and an 'Evaluate' button. Below the input field, it says 'You got 10 out of 10!'. The 'Feedbacks' section shows a list of feedback items, including '210 210 OK'.

Theory

Se da sirul $tab=[10, 47, 33, 29, 25, 18, 54]$ cu pivotul ales pe pozitia [3].
Primul element mai mare ca pivotul.
47
Daca nu exista se afiseaza valoarea pivotului.

Question

Parcurgeti sirul $tab=[87,121,108,210,60,273,152]$ de la stanga la dreapta si identificati primul element mai mare decat pivotul 3

Answers

210

You got 10 out of 10!

Feedbacks

210 210 OK

Fig.6.51. An implementation of identify the first element bigger than the pivot in quick sorting algorithm AGLO

The next AGLO aims for the student to know how to identify the first element smaller than the pivot. Also, if it does not exist, the student must know that the answer will represent the value of the pivot.

In the theory section, is presented a static example containing a table of seven integer elements. It is also emphasized that if there is no such element, the value of the pivot will be displayed.

In the question section, the student receives a table of randomly generated elements with a length between 5 and 8 and in which he must identify the first element bigger than the pivot. This notion is also important in understanding the steps of the quick sorting algorithm.

The answer is represented by the value of the element.

Theory

Se da sirul $tab=[10, 47, 33, 29, 25, 18, 54]$ cu pivotul ales pe pozitia [3].
Primul element mai mic decat pivotul.
18
Daca nu exista se afiseaza valoarea pivotului.

Question

Parcurgeti sirul $tab=[125,173,177,138,24]$ de la dreapta la stanga si identificati primul element mai mic decat pivotul 2

Answers

24

You got 10 out of 10! [Evaluate](#)

Fig.6.52. An implementation of identify the first element smaller than the pivot in quick sorting algorithm AGLO

The next AGLO aims for the student to know which is the first exchange that will be made in a given table with a given pivot.

Theory

Se da sirul $tab=[10, 47, 33, 29, 25, 18, 54]$ cu pivotul ales pe pozitia [3].
Interschimbare se va face intre elementele:
47 si 18

Question

Ce interschimbare se va face in sirul $tab=[73,99,33,253,288,58]$ avand pivotul 2

Answers

73 si 33

You got 10 out of 10! [Evaluate](#)

Fig.6.53. An implementation of the first exchange in Quick Searching Algorithm AGLO

In the theory section, is presented a static example containing a table of seven integer elements and the pivot with the index three, see Figure 6.41.

At each implementation of this AGLO generates an array of integers having the size in the range 5 - 10. In this case, the two previous applications are useful because the first change is made between the first element smaller than the pivot and the first element bigger than the pivot.

The last AGLO covers the steps of the quick sort algorithm.

Question

Care sunt versiunile intermediare ale sirului $tab=[157,253,202,299,66,32]$, in cadrul sortarii prin quick-sort?

Answers

```
s=1 d=2 m=1
32 66 157 299 202 253
s=3 d=5 m=4
32 66 157 202 299 253
s=4 d=5 m=4
32 66 157 202 253 299
```

You got 10 out of 10!

Fig.6.54. An implementation of the quick sorting algorithm steps AGLO

In this case, a table with a maximum of 10 integer elements is randomly generated. The student must write at each step which is the pivot and which are the limits of the substring on which it is worked, also the intermediate versions of the table.

6.1.10. Linked Lists AGLOs

Simply linked lists are dynamic data structures. Each node in the list contains, besides the useful information, the address of the next item. This organization only allows sequential access to the list items. To access the list, the address of the first element (called the head of the list) must be known; the following items are accessed through the list [66].

In order to cover in detail the addition and deletion in a list, we made eighteen AGLOs.

Inserting an element from a list can be done at the beginning, the end, or before, or after a given element. To perform these operations of addition to the list, we implemented several AGLOs.

The first AGLO deals with adding a node in an ordered linked list. The idea that should be emphasized in this example is that the addition to an ordered linked list is done in such a way that the list remains ordered.

In the theory section, the student receives an example, namely an ordered linked list of five elements in which another element is added. The list obtained is written in the form of a string, as this is the format in which the answer must be written.

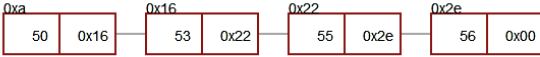
In the question section, an ordered linked list is randomly generated that can have between three and seven elements. The student receives this linked list in graphic form.

Theory

Adaugarea intr-o lista ordonata simplu inlantuita se face in asa fel incat lista sa ramana ordonata.
De exemplu daca se doreste adaugarea elementului {0x01,46,0x00}
in lista ordonata {0xa,48,0xe}->{0xe,58,0x1a}->{0x1a,61,0x22}->{0x22,70,0x26}->{0x26,82,0x2e}
Lista obtinuta va fi:
{0x01,46,0x0a}->{0xa,48,0xe}->{0xe,58,0x1a}->{0x1a,61,0x22}->{0x22,70,0x26}->{0x26,82,0x2e}

Question

Se da lista urmatoare:



Se va adauga elementul {0x01,59,0x00}.
Care este noua lista astfel obtinuta?

Answers

{0xa,50,0x16}->{0x16,53,0x22}->{0x22,55,0x2e}->{0x2e,56,0x01}->{0x01,59,0x00}

You got 10 out of 10!

Feedbacks

Adaugarea intr-o lista ordonata simplu inlantuita se face in asa fel incat lista sa ramana ordonata. In acest caz lista astfel obtinuta este: {0xa,50,0x16}->{0x16,53,0x22}->{0x22,55,0x2e}->{0x2e,56,0x01}->{0x01,59,0x00}.

Fig.6.55. An implementation of the adding a node in an ordered linked list AGLO

In Figure 6.55 you can see the graphic form that the linked list in the question has and the string shape that the student's answer has.

The theoretical notion which aims to add to an ordered linked list is resumed in the feedback section, and the correct answer is also represented.

The following AGLO aims to determine the position in which a node is added to an ordered linked list.

In the theory section, the student receives an example consisting of an ordered linked list of five elements and a node that would be added to it. Depending on the value of the information field of this node, the student must choose the position that the respective node will have in the list, namely at the beginning, at the end, or inside the list.

An ordered linked list is randomly generated that can have between three and seven elements and a node. The student must determine the position that the node will occupy so that the list remains in order. He has a choice of three options.

In the feedback section, this time the student receives only theoretical indications regarding the notion in question.

Theory

Adaugarea intr-o lista ordonata simplu inlantuita se face in asa fel incat lista sa ramana ordonata.
De exemplu daca se doreste adaugarea elementului {0x81,46,0x00}
in lista ordonata {0xa,48,0xe}->{0xe,58,0x1a}->{0x1a,61,0x22}->{0x22,70,0x26}->{0x26,82,0x2e}
raspunsul este: la inceput.

Question

Se da lista urmatoare:

0xa	41	0x12	0x12	46	0x16	0x16	52	0x22	0x22	62	0x2a	0x2a	71	0x00
-----	----	------	------	----	------	------	----	------	------	----	------	------	----	------

Unde se va adauga elementul {0x01,29,0x00}?

Variante de raspuns:

- la inceput
- la mijloc
- la sfarsit

Answers

la inceput

You got 10 out of 10! Evaluate

Feedbacks

Adaugarea intr-o lista ordonata simplu inlantuita se face in asa fel incat lista sa ramana ordonata. In acest caz adaugarea se face la inceput.

Fig.6.56. An implementation of the determining the position in which a node is added in an ordered linked list AGLO

Figure 6.56 represents an implementation of the AGLO that targets the position in which a node is added to an ordered linked list so that it remains ordered.

The next AGLO deals with a simpler but very important topic, which is the successor and the predecessor of a certain element in the linked list.

The theory section represents a random linked list of three elements and it is underlined that the predecessor of any element is the one located in the list before it. It is also important how an element of the linked list is represented. Each element is constructed in the form of a triplet so that when it is asked to name a certain element it represents the entire triplet consisting of the address of the element, the information of the element, and the address of the next element in the list.

To be able to practice this notion, the student receives a random list consisting of three to seven elements. Depending on the option that is randomly generated, the student must name the successor or predecessor of a certain element, also chosen at

random. In the exercise, however, the randomly selected element will never be the first or the last because in these cases inconveniences could have occurred such as the successor of the last element or the predecessor of the first element.

The answer consists of the required element.

In the feedback section, the student receives as information the correctly chosen node.

Theory

Intr-o lista predecesorul unui element este elementul situat in lista inaintea lui, iar succesorul este elementu aflat dupa el. In lista {0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00} Succesorul {0x1a,63,0x26} lui este {0x26,25,0x00} Predecesorul {0x1a,63,0x26} lui este {0x12,12,0x1a}

Question

Se da lista urmatoare:

0x16
0x1a

72
0x1a

→

0x1a
0x1e

42
0x1e

→

0x1e
0x00

28
0x00

Predecesorul elementului {0x1a,42,0x1e} este ...

Answers

{0x16,72,0x1a}

You got 10 out of 10! Evaluate

Feedbacks

{0x16,72,0x1a}

Fig.6.57. An implementation of the successor and the predecessor of an element in the linked list AGLO

Figure 6.57 shows an implementation of the AGLO in which the predecessor of an element is required.

The following AGLO deals with addition to the beginning or end of a random linked list.

As seen in Figure 6.58 for the addition, an option is first generated that will determine where to add the nod, at the beginning, or the end. An object of the form random linked list of three to seven elements and a node to be added are still randomly generated. A new object of form random linked list is then generated that will contain all the elements of the initial one and also the new element added on the corresponding position. The *toString* functions are necessary to be able to transmit to the student the correct information in the form of a character string.

In the theory section, the theory is mentioned, namely that the addition in a random linked list can be done at the beginning, at the end, or inside the list. The student is also presented with a static example to add at the beginning of a random list of three elements.

The expected answer in this AGLO consists of the new random linked list obtained after adding the element.

```

<scenario>
  <text>
A list of 3 - 7 random elements and a node will be generated that will be
added depending on the option: at the beginning, or the end.
  </text>
  <symbol name="optiune" type="integer">random(0,1,0);</symbol>
  <symbol name="locatie" type="string">v("optiune") ? "sfarsit" :
"inceput" ;</symbol>
  <symbol name="nNrElemente" type="integer">random(3,7,0);</symbol>
  <symbol name="elem" type="object">new LinkedListNode("0x01",
random(1,280,0), "0x00");</symbol>
  <symbol name="elem_str" type="string">v("elem").toString();</symbol>
  <symbol name="rll1" type="object">random_linked_list({"tabTypes" :
["pointer", "int"], "nNoOfElements" : v("nNrElemente"), "element": v("elem")})
</symbol>
  <symbol name="rll1_svg" type="string">v("rll1").toSVG();</symbol>
  <symbol name="rll2" type="string"> v("rll1").add(v("optiune"));
</symbol>
  <symbol name="rll2_str" type="string">v("rll2").toString();</symbol>
</scenario>

```

Fig.6.58. The scenario of the addition to the beginning/end of a random linked list AGLO

Theory

Adaugarea intr-o lista se poate face: la inceput, la sfarsit sau la mijloc.
De exemplu: Daca in lista {0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00}
dorim sa adaugam la inceput elementul {0x15,34,0x00}
lista obtinuta va fi {0x15,34,0x12}->{0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00}

Question

Se da lista urmatoare:

Adaugati la inceput elementul {0x01,235,0x00}.

Answers

{0x01,235,0x12}->{0x12,45,0x16}->{0x16,35,0x1e}->{0x1e,80,0x00}

You got 10 out of 10! Evaluate

Feedbacks

Elementul inserat in capul listei va avea campul de adresa urmator astfel incat sa poarte spre vechiul element cap de lista.
Elementul inserat in coada listei va fi referit de vechiul element coada.
In acest caz adaugarea se face astfel:

{0x01,235,0x12}->{0x12,45,0x16}->{0x16,35,0x1e}->{0x1e,80,0x00}

Fig.6.59. An implementation of the addition to the beginning/end of a random linked list AGLO

In the feedback section, see Figure 6.59, the student receives some theoretical information regarding the connections that are made as follows:

- the element inserted in the head of the linked list will have the next address field to point to the old element head of the list.
- the element inserted in the list queue will be referred to by the old queue element.

The student also receives the correct version of the linked list resulting from the addition of the new node.

The following AGLO aims to add a new node to a random linked list before or after a given element.

In the theory section, the student receives a static example in which the addition after a given element is illustrated.

To achieve this goal, one of the two options is randomly generated in the first place, namely, before or after.

De exemplu: Daca in lista {0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00}
dorim sa adaugam dupa elementul {0x1a,63,0x26} elementul {0x15,34,0x00}
lista obtinuta va fi {0x12,12,0x1a}->{0x1a,63,0x15}->{0x15,34,0x26}->{0x26,25,0x00}

Question

Se da lista urmatoare:

0x12	14	0x1a	0x1a	37	0x22	0x22	30	0x00
------	----	------	------	----	------	------	----	------

Dupa elementul {0x1a,37,0x22} adaugati elementul {0x01,157,0x00}.

Answers

{0x12,14,0x1a}->{0x1a,37,0x01}->{0x01,157,0x22}->{0x22,30,0x00}

Evaluate

You got 10 out of 10!

Fig.6.60. An implementation of the addition after/before an element in a random linked list AGLO

A random linked list is randomly generated. Its element to which the addition is made is also chosen randomly from its elements. The answer consists of the new list obtained after the addition see Figure 6.60.

The following AGLO aims to delete an element from a random linked list.

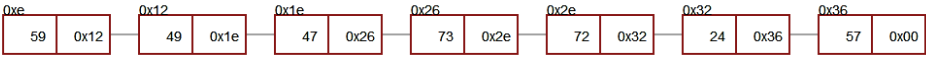
The theory section, see Figure 6.49, presents the notion of deleting a node from a random linked list. This is done by removing the named node. The student also receives a static example in which a node is removed from a random linked list.

In the question section, the student receives a random linked list of four to eight elements from which he is asked to delete a certain element. The node to be deleted is chosen randomly from the elements of the random linked list.

Theory

Stergerea dintr-o lista se face prin eliminarea nodului de sters. De exemplu: Daca in lista {0x12,12,0x1a}->{0x1a,63,0x15}->{0x15,34,0x26}->{0x26,25,0x00} dorim sa stergem elementul {0x15,34,0x26} lista obtinuta va fi {0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00}

Question



Stergeti din lista elementul {0x2e,72,0x32}

Answers

```
{0xe, 59, 0x12}->{0x12, 49, 0x1e}->{0x1e, 47, 0x26}->{0x26, 73, 0x32}->{0x32, 24, 0x36}->{0x36, 57, 0x00}
```

You got 10 out of 10!

Feedbacks

```
{0xe, 59, 0x12}->{0x12, 49, 0x1e}->{0x1e, 47, 0x26}->{0x26, 73, 0x32}->{0x32, 24, 0x36}->{0x36, 57, 0x00} {0xe, 59, 0x12}->{0x12, 49, 0x1e}->{0x1e, 47, 0x26}->{0x26, 73, 0x32}->{0x32, 24, 0x36}->{0x36, 57, 0x00}
```

Fig.6.61. An implementation of the delete an element from a random linked list AGLO

The answer consists of the random linked list obtained after deleting the targeted node.

The following AGLO aims to delete an element from the random linked list located before or after a given node.

As in the case of addition and the case of deletion, it may be required that the element have a certain location relative to a given node. We chose, and in this case, to use an option that was chosen randomly during implementation to determine the location before or after.

In the theory section, the student is presented with a static example in which an element located after a mentioned node is deleted from a random linked list, see Figure 6.62.

In the question section, it is represented graphically a random list consisting of four to eight nodes and it is required to delete an element located before or after a mentioned node. The mentioned node will never be the first or the last one to avoid the inconveniences that could occur, namely to delete the element located before the first element or the element located after the last element.

Theory

Stergerea dintr-o lista se face prin eliminarea nodului de sters.
 De exemplu: Daca in lista {0x12,12,0x1a}->{0x1a,63,0x15}->{0x15,34,0x26}->{0x26,25,0x00}
 dorim sa stergem elementul situat dupa {0x1a,63,0x15}
 lista obtinuta va fi {0x12,12,0x1a}->{0x1a,63,0x26}->{0x26,25,0x00}

Question



Stergeti elementul situat inainte de elementul {0x26,37,0x2e}

Answers

{0xe,33,0x26}->{0x26,37,0x2e}->{0x2e,70,0x00}

You got 10 out of 10! [Evaluate](#)

Feedbacks

{0xe,33,0x26}->{0x26,37,0x2e}->{0x2e,70,0x00} {0xe,33,0x26}->{0x26,37,0x2e}->{0x2e,70,0x00} OK

Fig.6.62. An implementation of the deletion after/before an element in a random linked list AGLO

The answer is represented by the new random linked list obtained after deleting the node.

In order to apply these notions, we also created AGLOs that have float or character data as information for the linked list nodes.

6.1.11. Double linked list AGLOs

A double-linked list contains an additional pointer to the previously linked list. The advantage is that this list can be traversed both forward and backward.

To acustom the student to the operations that can be performed on a double linked list, we made three types of AGLO:

- adding a node at the beginning, or the end;
- adding a node before or after a specified node;
- deleting a specified node.

We applied these types both to lists that have integers as information and to lists that have a float or a character as information. Thus the notion applied is the same, but the use case differs. The student can thus better master these things by repeating them.

```

<scenario>
  <symbol name="optiune" type="integer">random(0,1,0);</symbol>
  <symbol name="Locatie" type="string">v("optiune") ? "sfarsit" :
  "inceput" ;</symbol>
  <symbol name="nNrElemente" type="integer">random(3,7,0);</symbol>
  <symbol name="elem" type="object">new
  DoubleLinkedListNode("0x00","0x01", random(1,60,0), "0x00");</symbol>
  <symbol name="elem_str" type="string">v("elem").toString();</symbol>
  <symbol name="dll" type="object">random_double_linked_list({"tabTypes"
  : ["pointer","integer"], "nNoOfElements" : v("nNrElemente"), "element" :
  v("elem")});</symbol>
  <symbol name="dLLSVG" type="string">v("dll").toSVG();</symbol>
  <symbol name="dll1" type="object">
  v("dll").add(v("optiune"));</symbol>
  <symbol name="raspuns" type="string">v("dll1").toString();</symbol>
</scenario>

```

Fig.6.63.The scenario section of the adding a node to the beginning or end of a double linked list AGLO

The first AGLO deals with adding a node to the beginning or end of a list depending on the random value that the variable *optiune* will receive see Figure 6.63.

Theory

Exemplu:

Daca la inceputul listei {0x00,0x12,43,0x1a}->{0x12,0x1a,54,0x22}->{0x1a,0x22,20,0x01}->{0x22,0x01,23,0x00} se adauga elementul {0x00,0xa,21,0x12}

Atunci se obtine lista

{0x00,0xa,21,0x12}->{0xa,0x12,43,0x1a}->{0x12,0x1a,54,0x22}->{0x1a,0x22,20,0x01}->{0x22,0x01,23,0x00}

Question

Se da lista urmatoare:

0xa	0x00	72	0xe	→	0xe	0xa	16	0x1a	→	0x1a	0xe	70	0x00
-----	------	----	-----	---	-----	-----	----	------	---	------	-----	----	------

La inceput se va adauga elementul {0x1a,0x01,53,0x00}.

Answers

{0x00,0xa,72,0xe}->{0xa,0xe,16,0x1a}->{0xe,0x1a,70,0x01}->{0x1a,0x01,53,0x00}

Evaluate

You got 10 out of 10!

Fig.6.64. An implementation of the addition to the beginning/end of a double linked list AGLO

Depending on the value generated for the variable, the value of the option variable is still generated, having one of the beginning or end values. Its role is to indicate to the student which is the additional option to be performed.

It generates a node that will be added and the list in which the node will be added. The list is represented to the student in graphical form. The answer is made in the form of a string so that it can be written conveniently.

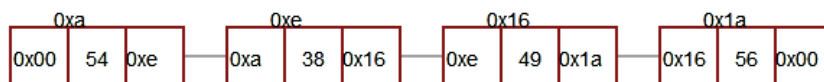
The theory section presents a static example in which a node is added to the beginning of a double list.

De exemplu:

Daca in lista $\{0x00,0x12,12,0x1a\} \rightarrow \{0x12,0x1a,63,0x26\} \rightarrow \{0x1a,0x26,25,0x00\}$ dorim sa adaugam dupa elementul $\{0x12,0x1a,63,0x26\}$ elementul $\{0x00,0x15,34,0x00\}$, lista obtinuta va fi $\{0x00,0x12,12,0x1a\} \rightarrow \{0x12,0x1a,63,0x15\} \rightarrow \{0x1a,0x15,34,0x26\} \rightarrow \{0x15,0x26,25,0x00\}$

Question

Se da lista urmatoare:



Inainte de elementul $\{0xa,0xe,38,0x16\}$ adaugati elementul $\{0x00,0x01,57,0x00\}$.

Answers

$\{0x01,0xa,54,0xe\} \rightarrow \{0x01,0xa,54,0xe\} \rightarrow \{0xa,0xe,38,0x16\} \rightarrow \{0xe,0x16,49,0x1a\} \rightarrow \{0x16,0x1a,56,0x00\}$

You got 10 out of 10!

Evaluate

Feedbacks

Elementul inserat trebuie sa pastreze legaturile catre celelalte elemente.

In acest caz adaugarea se face astfel:

$\{0x01,0xa,54,0xe\} \rightarrow \{0x01,0xa,54,0xe\} \rightarrow \{0xa,0xe,38,0x16\} \rightarrow \{0xe,0x16,49,0x1a\} \rightarrow \{0x16,0x1a,56,0x00\}$

Fig.6.65. An implementation of the addition after/before an element in a double linked list
AGLO

In the question section, the double list in which you want to add is represented in graphical form and the element to be added in the form of a string. We chose this

variant of representation because it is important for the student to have the graphic image of the connections between the nodes of the double linked list. Since it is necessary to write the answer, we considered the string to be the optimal variant.

As can be seen in Figure 6.64, the answer is the new double linked list obtained after the addition. The example shown above requires addition at the beginning of the list.

The following AGLO aims to add a node in a double linked list before or after a given element.

The theory section presents a static example that deals with the addition to a double linked list of a node after a given element.

The option for the location where the node will be added is randomly generated. Also, the element against which the location is established is randomly generated from the elements of the double linked list.

In the question section the student receives:

- the double linked list in which he must add represented graphically,
- the element to be added,
- the position in which it must be added,
- the element against which the position is established.

Theory

Stergerea dintr-o lista se face prin eliminarea nodului de sters.
 De exemplu:
 Daca in lista {0x00,0x12,e,0x1a}->{0x12,0x1a,h,0x15}->{0x1a,0x15,s,0x26}->{0x15,0x26,m,0x00}
 dorim sa stergem elementul {0x1a,0x15,s,0x26}
 lista obtinuta va fi {0x00,0x12,e,0x1a}->{0x12,0x1a,h,0x26}->{0x15,0x26,m,0x00}

Question

Stergeti din lista de mai sus elementul {0x22,0x26,r,0x2a}

Answers

{0x00,0x16,y,0x22}->{0x16,0x22,p,0x2a}->{0x22,0x2a,v,0x2e}->{0x2a,0x2e,k,0x32}->{0x2e,0x32,p,0x3a}->{0x32,0x3a,y,0x00}

You got 10 out of 10! Evaluate

Feedbacks

{0x00,0x16,y,0x22}->{0x16,0x22,p,0x2a}->{0x22,0x2a,v,0x2e}->{0x2a,0x2e,k,0x32}->{0x2e,0x32,p,0x3a}->{0x32,0x3a,y,0x00} {0x00,0x16,y,0x22}->{0x16,

Fig.6.66. An implementation of the delete an element from a double linked list AGLO

In the answer section, the student must write the new double linked list that is obtained after making the addition.

In the feedback section, it is mentioned that when adding a node to a double linked list it is important to consider the new connections between the nodes. Also here the student is shown the correct answer.

Figure 6.65 shows an implementation of adding before an item from a double linked list AGLO.

The following AGLO aims to delete a node from a double linked list.

A static example is represented in the theory section. This is the deletion of a node from a double list of four elements. The result is a new list that is obtained after removing the mentioned node.

In the question section, it is represented a double linked list of four to eight elements from which the student is asked to delete a certain node. The node to be deleted is chosen randomly from the elements of the double list.

The answer to this exercise is represented by the new list that is obtained after performing the operation of deleting the node.

Figure 6.66 shows the implementation of an AGLO that aims to delete an element from a double list in which the information is a character.

6.2. Models for Algorithm Analysis and Design Discipline

Regarding the design and analysis of algorithms, we chose to make some AGLOs about working with trees and graphs. Tree structures are used when a hierarchical organization is needed. Graphs are used to solve problems involving networks [67].

The generalized tree structure is important because it occurs frequently in practice, for example, family trees, or the structure of a book broken down into chapters, sections, paragraphs, and subparagraphs. Tree traversal algorithms are also very important and are recursive.

6.2.1. AGLOs regarding the notion of tree

We have developed nine AGLOs aimed at the correct knowledge and proper use of tree-specific notions, and three AGLOs aimed at traversing trees in preorder, postorder and inorder.

The first AGLO refers to the recognition of a tree.

To make this scenario we chose to generate three random structures: one tree and two that are not trees. The three structures will have the same number of nodes between 6 and 10, but will not necessarily have the same degree. The degree of the tree will be generated between 3 and 5, but the degree of graphs may vary depending on the random value that the symbol *var* will receive, see Figure 6.68. We chose this because we want to be able to generate as much variety as possible.

Structural objects will be initialized with graphic shapes using specific functions created toSVG(). These representations are very important in understanding the notions about trees.

In the theory section, the notion of a tree is presented. A tree is a finite set of elements in which:

- i) there is a special node called the root;
 ii) the rest of the nodes are partitioned into $n > 0$ sets M_1, \dots, M_n where each set represents a tree.
 M_1, \dots, M_n is the root subtrees.

```

|scenario>
. . .
<symbol name="n" type="integer">random(6,10,0);</symbol>
<symbol name="d" type="integer">random(3,5,0);</symbol>
<symbol name="var" type="integer">random(0,1,0)</symbol>
<symbol name="tree" type="tree"> random_tree( {"nNoOfNodes" :
v("n"), "nDegree":v("d")}); </symbol>
<symbol name="graph1" type="graph">random_graph({"nNoOfNodes" :
v("n"), "nDegree":v("d")+v("var")}); </symbol>
<symbol name="graph2" type="graph">random_graph({"nNoOfNodes" :
v("n"), "nDegree":v("d")-v("var")}); </symbol>
<symbol name="treesvg" type="string">v("tree").toSVG(); </symbol>
<symbol name="graph1svg" type="string">v("graph1").toSVG(); </symbol>
<symbol name="graph2svg" type="string">v("graph2").toSVG(); </symbol>
</scenario>

```

Fig.6.67. The scenario of the recognition of a tree AGLO

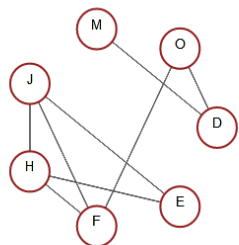
Theory

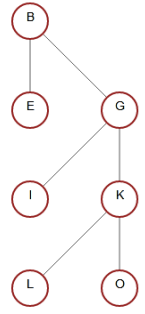
Un arbore este o multime finita elemente in care:
 i) exista un nod special numit radacina;
 ii) restul nodurilor sunt partitionate in $n > 0$ multimi M_1, \dots, M_n unde fiecare multime reprezinta un arbore.
 M_1, \dots, M_n sunt subarborii radacinii.

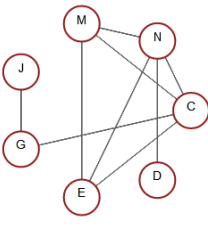
Question

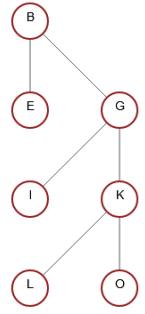
Identificati care din structurile desenate mai jos corespunde definitiei de arbore:

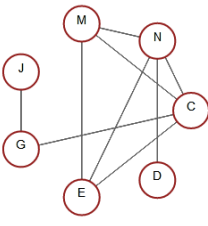
Answers











Correct!

Feedbacks

In prima si a treia structura exista arce ce determina o legatura intre doi subarbori diferiti deci structurile nu sunt arbori.

Evaluate

Fig.6.68. An implementation of the recognition of a tree AGLO

In the question section, the student is asked to choose from the represented structures the one that is a tree.

In the answer section, the student sees the representations of the three structures and can choose the right one. At each implementation, the second structure is the one that can be a tree, only each time it will have a different representation.

In the feedback section, it is explained to the student that in the first and third structures there are arches that determine a connection between two different subtrees so the structures are not trees.

Figure 6.68 shows an implementation of the AGLO that aims to recognize a tree with the correct answer.

Question

Care din nodurile structurii ar putea avea rol de radacina?

```

graph TD
    B((B)) --- E((E))
    B --- G((G))
    G --- J((J))
    G --- K((K))
  
```

Answers

doar B
 B si J
 doar G
 J si K
 Oricare nod.
 Correct!

Evaluate

Feedbacks

In structura data orice nod ar putea fi radacina.

Fig.6.69. An implementation of the recognition the tree root AGLO

The following AGLO refers to recognizing the root of a tree.

In this scenario, we chose to randomly generate a tree with five nodes that can have degree 1 or 2. Also based on the node keys we will initiate some variables that will represent response variants.

In the question section, the student is asked to choose from the answer options below the one that denotes which of the nodes of the structure could have the role of the root.

There are five answer options in the answer section. Three of the answers are the same in each instance, and two are variable depending on the randomly generated options. The only correct option is the last one.

In the feedback section, it is explained that in the represented structure any node could be chosen as the root.

Figure 6.69 shows an implementation of the AGLO that aims to recognize a tree root with the correct answer.

Theory

Într-un arbore nivelul maxim ne da înaltimea sa.

Question

Scrieti care este înaltimea arborelui din figura de mai jos:

```

graph TD
    B((B)) --- C((C))
    B --- E((E))
    E --- G((G))
    E --- H((H))
    H --- K((K))
    H --- M((M))
    I((I)) --- M
    I --- N((N))

```

Answers

4

You got 10 out of 10! Evaluate

Feedbacks

Intrucat avem 4 nivele, înaltimea arborelui este tot 4.

4
4
OK

Fig.6.70. An implementation of the calculation of a tree height AGLO

The following AGLO aims to calculate the height of a tree.

In this case, a tree with 6 to 10 nodes is randomly generated, which can have a degree between 2 and 5. The graphic representation in the SVG format of the tree is also generated. The height is calculated by calling the specific method that we performed `tree.height()`.

In the theory section, it is mentioned that in a tree the maximum level gives us its height.

In the question section, the student is informed that it is desired to calculate the height of the represented tree.

In answer, he has a certain natural number to write which represents the number of levels of the tree, i.e. its height.

In the feedback section, it is explained to the student that the number of levels represents the height of the tree. Also, the answer given by the student is compared to the one expected to be received.

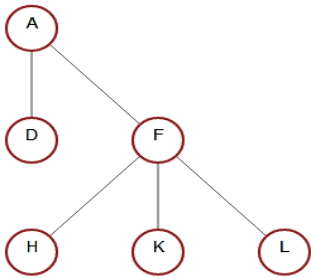
An implementation of the AGLO that aims to calculate the height of a tree with the correct answer is shown in Figure 6.70.

Theory

Într-un arbore, nivelurile sunt definite astfel:
 rădăcina formează nivelul 1,
 fiii săi formeaza nivelul 2,
 și așa mai departe,
 fiii nodurilor de pe nivelul n formează nivelul n+1.

Question

Scrieti pe cate un rand fiecare nivel din arborele de mai jos, punand cate un spatiu intre nodurile de pe acelasi nivel:



```

graph TD
  A((A)) --- D((D))
  A --- F((F))
  F --- H((H))
  F --- K((K))
  F --- L((L))
  
```

Answers

A
 D F
 H K L

You got 10 out of 10! Evaluate

Feedbacks

Se ia pe rand fiecare nivel si se scriu nodurile.

A A OK
 OK

 D D OK
 F F OK

Fig.6.71. An implementation of the distribution of nodes of a tree by levels AGLO

The following AGLO aims to determine all the nodes that are on each level in the tree.

To make this scenario we chose to randomly generate a tree with 6 to 10 nodes. To represent in string format the nodes that are on each level of the tree, we realized the tree-levels function.

In the theory section, the student is reminded that in a tree, the levels are defined as follows: the root forms level 1, its sons form level 2, and so on, the sons of nodes on level n form level $n + 1$.

Theory

Într-un arbore prin gradul unui nod intelegem numarul de fii al acestuia.

Question

Scriti care este gradul nodului S din figura de mai jos:

```

graph TD
    A((A)) --- D((D))
    A --- G((G))
    G --- I((I))
    G --- L((L))
    G --- O((O))
    G --- R((R))
    G --- S((S))
    S --- U((U))
    S --- X((X))
        
```

Answers

Evaluate

You got 0 out of 10!

Feedbacks

Pentru a determina gradul unui nod pur si simplu numaram cati fii are acel nod.

5 2 NOT OK

Fig.6.72. An implementation of the degree of a node of a tree AGLO

In the question section you will find its statement, namely: Write each level in the tree below one by one, putting a space between the nodes on the same level. This statement also indicates how the correct answer should be written.

The answer thus includes all the nodes of the tree distributed on levels, as can be seen in Figure 6.71.

The feedback section consists of two parts:

- a static one that includes an explanation of the exercise, namely that for the correct solution each level is taken in turn and the nodes are written;
- a dynamic in which the answer given by the student is compared with the correct answer.

The following AGLO refers to determining the degree of a node.

To make this scenario we randomly generated three numbers like this; the first located between 6 and 12 represents the number of tree nodes, the second located between 2 and 5 represents the degree of the tree, and the third located in the interval 0 and the first represents the node whose degree we want to determine. Based on the first two numbers, a tree is randomly generated and its representation.

In the theory section, it is mentioned that in a tree by the degree of a node we understand its number of sons.

In the question section, the student is asked: what is the degree of the node chosen randomly from the nodes of the tree.

The answer consists of a single number that represents the degree of the node. Figure 6.72 shows an implementation with the wrong answer. In this case, the student can receive either the correct score or nothing, as he has to write a single number that can be right or wrong.

In the feedback section, the student is reminded that he had to count the sons of the respective node.

The following AGLO refers to determining the degree of a tree.

In this scenario, a tree with 6-15 nodes, and a degree between 2 and 5 are randomly generated. The degree of a tree is given by the maximum degree of its nodes.

In the question section, the student is asked to calculate the degree of the represented tree.

The answer also consists in this case of a single number, which means that it can be either right or wrong.

The feedback emphasizes the idea in theory, namely that we must find the node with the most sons, and the student's answer is compared with the expected one.

The following AGLO refers to determining the degrees of all tree nodes.

A tree is also randomly generated for this scenario.

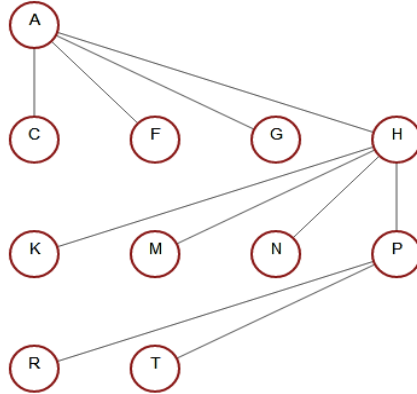
The theory section specifies that in a tree by the degree of a node we understand the number of its sons.

Theory

Gradul unui arbore este dat de gradul maxim al nodurilor acestuia.

Question

Scrieti care este gradul arborelui din figura de mai jos:



Answers

You got 10 out of 10!

Evaluate

Feedbacks

Pentru a determina gradul unui arbore trebuie sa gasim nodul cu cel mai multi fii sinonim cu gradul cel mai mare.

4 4 OK

Fig.6.73. An implementation of the degree of a tree AGLO

As can be seen in Figure 6.74, the question is given to the student and the details about who should give the correct answer.

According to the instructions received in the answer section, the student should write on each row a node starting from the root followed by its degree. Thus the answer has as many lines as the number of nodes in the tree.

Theory

Într-un arbore prin gradul unui nod intelegem numarul de fii al acestuia.

Question

Scrieti pe cate o linie, sub forma nod grad (separat prin spatiu), care este gradul nodurilor in ordine din figura de mai jos:

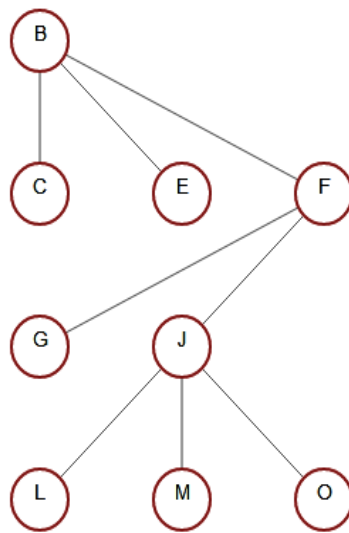


Fig.6.74. The theory and the question of the degree of all nodes of a tree AGLO

In the feedback section, it is pointed out that to determine the degree of a node we simply count how many sons that node has. Also to determine the degree of all nodes in the tree we apply this procedure to all nodes in the tree, and the terminal nodes have degree 0. The student's answer is also compared to the correct answer expected, for reasons of space in Figure 6.75 we did not leave the whole comparison.

The following AGLO aims to represent all pairs of form the node and father node pairs in a tree.

The notions concerned refer to the knowledge by the student of the fact that a node A is a direct son of another node B if it is located on the level immediately following node B and there is an edge between A and B. A node A is the parent of another node B if it is located on the level immediately above node B and there is an edge between A and B.

Answers

```
B 3
C 0
E 0
F 2
G 0
J 3
```

You got 10 out of 10!

Feedbacks

Pentru a determina gradul unui nod pur si simplu numaram cati fii are acel nod.
 Aplicam acest procedeu pentru toate nodurile din arbore.
 Nodurile terminale au gradul 0 (zero).

```
B B OK
3 3 OK
C C OK
0 0 OK
```

Fig.6.75. The answer and the feedback of the degree of all nodes of a tree AGLO

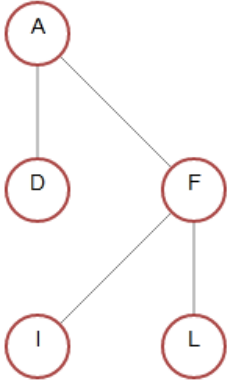
```
<scenario>
. . .
<symbol name="n" type="integer">random(4,8,0);</symbol>
<symbol name="d" type="integer">random(2,4,0);</symbol>
<symbol name="tree" type="tree"> random_tree({"nNoOfNodes": v("n"),
"nDegree":v("d")});</symbol>
<symbol name="treesvg" type="string">v("tree").toSVG(); </symbol>
<symbol name="answer" type="string">v("tree").getChildParent();
</symbol>
</scenario>
```

Fig.6.76. The scenario of the pears node – parent node of a tree AGLO

To perform this exercise in the scenario section, Figure 6.76, a tree is randomly generated with the number of nodes between 4 and 8 and the degree between 2 and 4. The expected answer is the result returned by the function `getChildParent()`.

Question

Scriti pe cate un rand fiecare pereche de noduri fiu-tata.



```

graph TD
  A((A)) --- D((D))
  A --- F((F))
  F --- I((I))
  F --- L((L))
  
```

Answers

D A
F A
I F
L F

You got 10 out of 10!

[Evaluate](#)

Fig.6.77. The question and the answer of the pairs node – parent node of a tree AGLO

In the question section, the student is asked to write, for each node of the tree, one by one, the paired node - parent node. The structure of the answer can be seen in Figure 6.77.

In the feedback section, it is emphasized that each node had to be taken in turn and its parent was written next to it. Also, the only node that does not have a father, the root node, should not appear as the first element in any row.

The following AGLO aims to determine the root nodes of each subtree of a node.

To realize the scenario aiming at the exercise whose instantiation can be seen in Figure 6.78, a random tree structure with 6 - 10 nodes and degree 2-4 is generated. An internal node is also chosen at random.

It is desired to determine the root nodes of each subtree of the chosen node. The theory section mentions that this means determining the sons of that node.

The feedback consists of two parts:

- one part that explains what was to be done, namely take the nodes below the indicated node one by one and list them,

- one part in which the answer received is compared with the expected answer unit by unit.

Question

Scrieti pe cate un spatiu intre ele nodurile radacina ale fiecarui subarbore al nodului I.

```

graph TD
    A((A)) --- B((B))
    A --- D((D))
    A --- F((F))
    A --- I((I))
    I --- L((L))
    I --- N((N))
    I --- O((O))
    I --- R((R))

```

Answers

L N O R

You got 10 out of 10!

[Evaluate](#)

Feedbacks

Se iau pe rand nodurile de sub nodul indicat si se listeaza.

```

L L OK
N N OK
O O OK
R R OK

```

Fig.6.78. An implementation of determination the root nodes of each subtree of a node AGLO

The next three AGLOs refer to crossing a tree. By crossing the tree we mean visiting each node and processing its specific information. For a given tree, corresponding to a certain application, a certain traversal order is required. The programs use systematic shaft traversal algorithms implemented in the form of procedures. The navigation possibilities are inorder, preorder, and postorder.

The first of the mentioned AGLOs aims at traversing a tree in order.

```

<scenario>
  . . .
  <symbol name="n" type="integer">random(6,10,0);</symbol>
  <symbol name="d" type="integer">random(2,5,0);</symbol>
  <symbol name="tree" type="tree"> random_tree( {"nNoOfNodes" :
v("n"), "nDegree":v("d")});</symbol>
  <symbol name="treesvg" type="string">v("tree").toSVG();</symbol>
  <symbol name="answer" type="string">inorder(v("tree"));</symbol>
</scenario>

```

Fig.6.79. The scenario of traversing the nodes of a tree in inorder AGLO

As can be seen in Figure 6.79 to achieve this scenario, two numbers are randomly generated, one in the range 6, 10 and represents the number of tree nodes, and one in the range 2, 5 and represents the degree of the tree. A tree with the respective dimensions is randomly generated, as well as its graphical representation. In order to obtain the desired crossing, the *inorder()* function is called.

In the theory section, it is presented the fact that to go through a tree in inorder, the following steps are performed:

1. Go through the left subtree,
2. Visit the root,
3. Go through the right subtree.

In the question section, the student receives the graphic shape of the tree to be traversed.

In the answer section, all tree nodes were expected to be written in inorder.

The feedback section highlights how the browsing is done, namely the left subtree, root, and right subtree. Next, the answer given is compared with the correct answer expected.

A representation of an implementation of the traversal in inorder is shown in Figure 6.80.

The next AGLO aims at traversing a tree in preorder.

To achieve this scenario, whose implementation can be seen in Figure 6.82, a tree with between 6 and 10 nodes, and with a degree between 2 and 5 were randomly generated.

In the theory section, the steps of the preorder traversal algorithm are presented:

1. Visit the root,
2. Go through the left subtree,
3. Go through the right subtree.

In the answer section, all tree nodes were expected to be written in preorder.

The randomly generated tree is graphically represented in the question section.

The feedback section highlights how the traversal is done, namely root, left subtree, and right subtree, and the answer given by the student is compared with the correct answer expected.

Theory

Parcurgerea în inordine a unui arbore se face astfel:

1. Parcurgeți subarborele din stânga,
2. Vizitați rădăcina,
3. Parcurgeți subarborele drept.

Answers

C B I F K M

You got 10 out of 10!

Question

Scrieti parcurgerea in preordine a arborelui (pe un singur rand cu cate un spatiu intre noduri):

```

    graph TD
      B((B)) --- C((C))
      B --- F((F))
      F --- I((I))
      F --- K((K))
      F --- M((M))
    
```

Answers

Parcurgerea se face astfel: fiu stang, radacina, fiu drept.

C C OK
 B B OK
 I I OK
 F F OK
 K K OK
 M M OK

Fig.6.80. An implementation of traversing the nodes of a tree in inorder AGLO

Theory

Parcurgerea în preordine a unui arbore e face astfel:

1. Vizitați rădăcina,
2. Parcurgeți subarborele din stânga,
3. Parcurgeți subarborele drept.

Answers

A C E G H J K

You got 10 out of 10!

Question

Scrieti parcurgerea in preordine a arborelui (pe un singur rand cu cate un spatiu intre noduri):

```

    graph TD
      A((A)) --- C((C))
      A --- E((E))
      A --- G((G))
      G --- H((H))
      G --- J((J))
      G --- K((K))
    
```

Answers

Parcurgerea se face astfel: radacina, fiu stang, fiu drept.

A A OK
 C C OK
 E E OK
 G G OK
 H H OK
 J J OK
 K K OK

Fig.6.81. An implementation of traversing the nodes of a tree in preorder AGLO

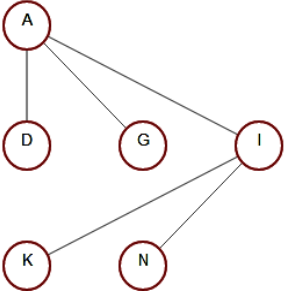
Theory	Answers
Parcurgerea în postordine a unui arbore e face astfel: 1. Parcurgeți subarborele din stânga, 2. Parcurgeți subarborele drept, 3. Vizitați rădăcina.	D G K N I A You got 10 out of 10! Evaluate
Question	Feedbacks
Scrieti parcurgerea in postordine a arborelui (pe un singur rand cu cate un spatiu intre noduri):  <pre> graph TD A((A)) --- D((D)) A --- G((G)) G --- I((I)) I --- K((K)) I --- N((N)) </pre>	Parcurgerea se face astfel: fiu stang, fiu drept, radacina. D D OK G G OK K K OK N N OK I I OK A A OK

Fig.6.82. An implementation of traversing the nodes of a tree in post order AGLO

The next AGLO aims at traversing a tree in post order. The structure of the AGLO is based on the same ideas as the other crossings. Figure 6.82 shows an implementation of the AGLO that aims to traverse a tree in post order.

In the theory section is presented the way in which the post order is traversed, namely: the left subtree, the right subtree, and finally the root.

In the questions section, a tree with a maximum of ten nodes is represented in SVG format. The student is asked to go through the respective graph in post order.

In the answer section, all tree nodes were expected to be written in post order as shown in Figure 6.82.

In the feedback section, is reminded the correct order for this traversal: left subtree, right subtree, and root, and the correct answer is compared with the given answer.

6.2.2. AGLOs regarding the notion of graph

In the next group of objects, the first AGLO refers to recognize a graph versus a tree.

To make this scenario we chose to generate three random structures: one graph and two trees. The number of nodes is chosen randomly between 6 and 10. The generated structures may or may not have the same number of nodes, depending on the value that the variable *var* will have. Thus if the variable has the value zero then all the structures have the same number of nodes, if the variable has the value 1 then each structure will have a different number of nodes.

Structural objects will be initialized with graphic shapes using specific functions created toSVG(). These representations are very important in understanding the notions.

```
<scenario>
  <text>
    Se genereaza trei structuri aleatoare: una graf si doi arbori.
    Studentul trebuie sa recunoasca un graf fata de un arbore.
  </text>
  <symbol name="n" type="integer">random(6,10,0);</symbol>
  <symbol name="d" type="integer">random(3,5,0);</symbol>
  <symbol name="var" type="integer">random(0,1,0)</symbol>
  <symbol name="tree1" type="tree">
    random_tree({"nNoOfNodes" : v("n"), "nDegree":v("d")+v("var")});</symbol>
  <symbol name="graph" type="graph">
    random_graph({"nNoOfNodes" : v("n"), "nDegree":v("d")});</symbol>
  <symbol name="tree2" type="tree">
    random_tree({"nNoOfNodes" : v("n"), "nDegree":v("d")-v("var")});</symbol>
  <symbol name="tree1svg" type="string">v("tree1").toSVG();</symbol>
  <symbol name="graphsvg" type="string">v("graph").toSVG();</symbol>
  <symbol name="tree2svg" type="string">v("tree2").toSVG();</symbol>
</scenario>
```

Fig.6.83. The scenario of the recognition of a graph AGLO

In the theory section it is reminded that a graph is an ordered pair of sets, denoted $G = (X, U)$. X is a finite and nonempty set of elements called nodes or vertices, and U is a set of pairs (ordered or unordered) of elements in X called edges (if they are unordered pairs) or arcs (if they are ordered pairs) [67].

In the question section, the student is asked to choose from the represented structures the one that is a graph.

In the answer section, the student sees the representations of the three structures and can choose the right one. At each implementation, the second structure is the one that can be a graph, only each time it will have a different representation.

In the feedback section, it is explained to the student that in the first and third structures are trees, and the second is a graph.

The following AGLO refers to calculate the maximum degree of a graph.

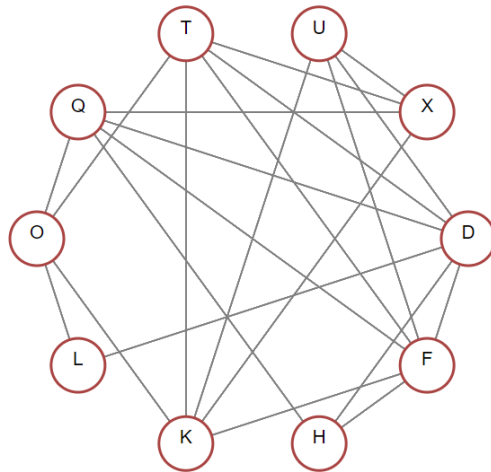
In this scenario, we chose to randomly generate a graph with a number of nodes randomly generated between 6 and 10. We will also initialize a variable with the representation of the graph and a response variable containing the maximum degree of the graph.

Theory

The maximum degree of a graph G is the maximum degree of its nodes.

Question

Determine the maximum degree of the graph below:



Answers

You got 10 out of 10!

[Evaluate](#)

Feedbacks

The degree of a node is given by the number of edges adjacent to it.

6 6 OK

Fig.6.84. An implementation of the degree of a graph AGLO

In the theory section, it is presented that the maximum degree of a graph G is the maximum degree of its nodes.

In the question section, the student is asked to determine the maximum degree of the represented graph.

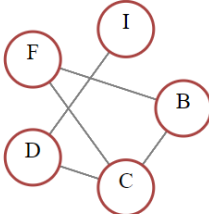
In the feedback section the student is reminded of the way in which the degree of a node is calculated, namely this represents the number of edges adjacent to the node.

Figure 6.84 shows an implementation of the AGLO that aims to calculate the degree of a graph. In this case, a graph with ten nodes was generated, whose maximum degree is 6. We chose to give the correct answer. Thus, after the automatic evaluation, the grade is maximum.

The next AGLO aims to determine the minimum path from the first node to all other nodes with the help of Dijkstra's algorithm.

Theory

Consider the undirected graph below:



Having as the cost matrix:

```
00 42 00 14 00
42 00 13 43 00
00 13 00 00 64
14 43 00 00 00
00 00 64 00 00
```

In order to determine the minimum path that can be traveled from the first node to all the others with the help of Dijkstra's algorithm, the following steps are followed:

```
0 42 1000 14 1000
0 42 55 14 1000
0 42 55 14 119
0 42 55 14 119
0 42 55 14 119
```

Indications:

- 1.The order of the nodes is considered alphabetical.
- 2.Path length values are calculated in an array.
- 3.The value 1000 is considered the infinite path.

In this situation, the distances from the first node of the above graph to the other nodes, in order, are:

```
0 42 55 14 119
```

Fig.6.85. The theory section of the minimum path from the first node to all other nodes AGLOs

In the scenario section, two integers are randomly generated,
 - one from 6 to 10 and represents the number of nodes of the graph;
 - one in the range [3; 5] and represents the maximum degree that a node can have.

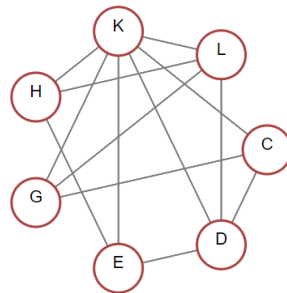
A graph is then generated based on these values. In order to have all the necessary elements for the targeted learning objective, the graphical representation

of the graph, the cost matrix associated with the graph and the minimum paths from the first node to all other nodes are generated using Dijkstra's algorithm.

An example is presented in the theory section so that the student can recapitulate all the necessary theoretical part see Figure 6.85. A five-node graph is presented to which a cost matrix is attached. All the steps that Dijkstra's algorithm follows are presented below. The answer to the exercise is given by the best version generated by the algorithm. In addition, the student receives some indications, namely that the order of the nodes is considered alphabetical; there are situations in which there is no arc between two nodes, the infinite path will be symbolized by the value 1000.

Question

Consider the undirected graph below:



Having as the cost matrix:

```
00 50 00 19 00 64 00
50 00 45 00 00 38 25
00 45 00 00 19 22 00
19 00 00 00 00 21 18
00 00 19 00 00 63 42
64 38 22 21 63 00 38
00 25 00 18 42 38 00
```

The distances from the first node of the graph to the other nodes, in order, are:

Fig.6.86. The question section of the minimum path from the first node to all other nodes
AGLOs

In the questions section, a graph is represented together with the related cost matrix. The student is asked to determine the minimum path from the first node to all the others. Figure 6 shows a question that includes a graph with seven nodes, the first node being C.

In the answer section, the student must write the minimum lengths of the paths from node C to the others. Figure 6 shows the response and feedback sections. The answer is written in the form of a table of values.

In the feedback section, the student is reminded that he had to calculate the length of the paths from the first node to all the others, and by applying the algorithm at each step, the minimum distance found is optimized. The correct answer was also given.

The screenshot shows a quiz interface with two main sections: 'Answers' and 'Feedbacks'.

Answers section: A light blue header is followed by a text input field containing the sequence of numbers: 0 50 62 19 79 40 37. Below the input field, it says 'You got 10 out of 10!' and there is a blue 'Evaluate' button with a checkmark icon.

Feedbacks section: A light yellow header is followed by a text area containing the instruction: 'Calculate the length of the roads from the first node to all the others, at each step of the algorithm optimizing with the minimum distance found.' Below this, there is a list of feedback items, each showing a node ID and the word 'OK':

- 0 0 OK
- 50 50 OK
- 62 62 OK
- 19 19 OK
- 79 79 OK
- 40 40 OK
- 37 37 OK

Fig.6.87. The answers and feedbacks sections of the minimum path from the first node to all other nodes AGLOs

The following AGLO aims to determine the adjacency matrix attached to the graph.

For this exercise in the scenario section two integers are randomly generated:

- the first represents the number of nodes of the graph and is generated in the range [6; 10];
- the second represents the maximum degree of one node and is generated in interval [3; 5].

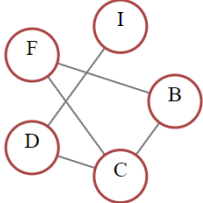
Based on these, a graph is randomly generated. For the representation, the graphical representation in SVG format is generated, and for the answer, the adjacency matrix corresponding to the graph is calculated.

In the theory section, is presented the way in which the adjacency matrix is constructed, namely this is a quadratic matrix whose elements have the values 1 or 0 depending on whether or not there is an arc between the two nodes. In addition, because the graph is undirected the matrix will be symmetrical. Below is a five-node graph and its adjacent matrix.

In the question section, the student is asked to construct the adjacency matrix of the represented graph. Figure 6.88 shows an example in which a graph with six nodes is generated: D, E, G, H, K, and N, each having a maximum degree of 3.

Theory

Let $G = (V, M)$ be an undirected graph with n vertices (V) and m edges (M). The adjacency matrix, associated with the graph G , is a quadratic matrix of order n , with the elements defined as follows:
 $-a[i,j]=1$, if $[i,j]$ belongs to M ; $-a[i,j]=0$, if $[i,j]$ does not belongs to M .
 For example, for the graph:



The adjacency matrix, associated with is:

```

0 1 0 1 0
1 0 1 1 0
0 1 0 0 1
1 1 0 0 0
0 0 1 0 0

```

Question

Determine the adjacency matrix, associated with the graph below:

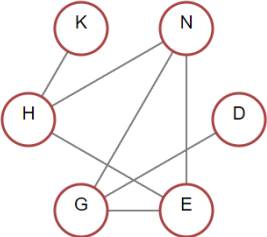


Fig.6. 88. The theory and question sections of the adjacency matrix AGLO

Figure 6 shows a correct answer together with the corresponding feedback. The answer given by the student must respect the representation of the matrix.

In the feedbacks section, the student is reminded that he had to construct a quadratic matrix whose elements are 0 or 1 depending on the existence of the edge $[i, j]$ in the given graph. In addition, the answer given by the student is compared with the correctly calculated answer.

Answers

```

0 0 1 0 0 0
0 0 1 1 0 1
1 1 0 0 0 1
0 1 0 0 1 1
0 0 0 1 0 0
0 1 1 1 0 0

```

Evaluate

You got 10 out of 10!

Feedbacks

You had to make a quadratic matrix whose elements are 0 or 1 depending on the existence of the edge [i, j].

```

0 0 OK
0 0 OK
1 1 OK
0 0 OK
0 0 OK
0 0 OK
0 0 OK
OK

0 0 OK
0 0 OK
1 1 OK
1 1 OK
0 0 OK
1 1 OK
OK

1 1 OK
1 1 OK
0 0 OK
0 0 OK
0 0 OK
1 1 OK
OK

```

Fig.6. 89. The answers and feedbacks sections of the adjacency matrix AGLO

The following AGLO aims to determine the adjacency structure attached to the graph.

For this exercise in the scenario section two integers are randomly generated:

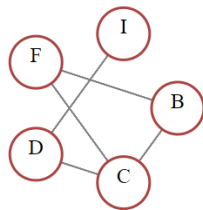
- the first represents the number of nodes of the graph and is generated from 6 to 10;
- the second represents the maximum degree of one node and is generated from 3 to 5.

Based on these, a graph is randomly generated. For the representation, the graphical representation in SVG format is generated, and for the answer, the adjacency matrix corresponding to the graph is calculated.

In the theory section, is presented the way in which the adjacency matrix is constructed, namely on each row is written a node followed by all the nodes with which it is adjacent in alphabetical order.

Theory

For example, for the graph:



The adjacency structure, associated with is:

```

B C F
C B D F
D C M
F B C
M D
  
```

Question

Determine the adjacency structure, associated with the graph below:

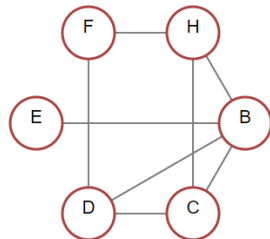


Fig.6. 90. The theory and question sections of the adjacency structure AGLO

In the theory section is represented a graph of five nodes and the corresponding adjacent structure. Figure 6.90 captures the representation of the theory that appears at each implementation of the AGLO.

In the question section is represented the graph randomly generated in the scenario. The student is asked to construct the adjacent structure related to the graph. Figure 6.90 shows an example in which a graph of six nodes B, C, D, E, F and H has been generated.

Answers

B C D E H
C B D H
D B C F
E B
F D H

You got 10 out of 10!

Evaluate

Fig.6. 91. The answers and feedbacks sections of the adjacency matrix AGLO

In the answers section, the student is expected to write on a line in alphabetical order each node followed by the nodes with which it is adjacent. Figure 6.91 shows a correct response to an implementation of this AGLO.

6.3. Models for Operating Systems Discipline

Linux is a family of Unix-type operating systems that use the kernel. Linux can be installed on a wide variety of hardware, from mobile phones, tablets, video consoles, to personal computers to supercomputers, which is why it is widely used in some international companies. Linux is best known for its use as a server. In recent years, Linux has begun to become increasingly popular due to distributions such as Ubuntu, open SUSE, Fedora, as well as the advent of notebooks and the new generation of smartphones running an embedded version of Linux.

These are sufficient reasons for us to approach applications basic notions in this field.

6.3.1. Commands for directories AGLOs

The first set of AGLOs aims to exercise the commands for working with directories.

The first AGLO aims to use the correct command to create a directory.

```
<scenario>
  <text>
A directory name is randomly generated.
  </text>
  <symbol name="folder" type="object">new Folder(1,null)</symbol>
  <symbol name="director" type="string">v("folder").toString();</symbol>
  <symbol name="comanda" type="string">"mkdir "+v("director");</symbol>
</scenario>
```

Fig.6.92. The scenario for the command to create a directory AGLO

Figure 6.92 shows the scenario of the AGLO which aims to create a directory using the *mkdir* command. The name of the directory is chosen at random from a

series of names that represent letters of the Greek alphabet. The correct answer is retained in the *comanda* symbol.

The theory section presents a static example, namely how to create the *alfa* directory.

In the question section the student receives the statement of the question in which he is told what command to use and what name the directory has.

In the answer section the student writes the command line necessary to fulfill the requirement from the previous section.

The screenshot displays an AGLO interface with four main sections:

- Theory:** A green header with the text "Example of creating a folder named alfa." and a code box containing `mkdir alfa`.
- Question:** A red header with the text "Use the mkdir command to create the folder with the name Papa18."
- Answers:** A blue header with a text input field containing `mkdir Papa18`. Below the input, it says "You got 10 out of 10!" and there is an "Evaluate" button.
- Feedbacks:** A yellow header with the text "Basically, you had to write a single command to create the folder." and a code box containing:

```
mkdir mkdir OK
Papa18 Papa18 OK
```

Fig.6.93. An instance of the command to create a directory AGLO

Figure 6.93 shows an instantiation of the AGLO aimed at creating the directory named Papa18. In this instant, we put a correct answer, and after evaluating the answer, the grade that the application automatically gives is maximum.

The following AGLO aims to create a directory hierarchy using the *-p* option. In this case, each folder does not have to be created individually.

```

<scenario>
  <text>
A random tree is generated containing various names of the number of nodes n.
It is necessary to represent in text format on levels but indented. The
answer involves calculating all the paths in the tree. It is necessary to
create a function that returns all paths in the tree as a string containing
several lines separated by \n.
  </text>
  <symbol name="n" type="integer">random(5,8,0);</symbol>
  <symbol name="ft" type="FolderTree">new FolderTree({"nNoOfNodes" :
v("n")});</symbol>
  <symbol name="st" type="string">v("ft").toString();</symbol>
  <symbol name="comenzi" type="string">v("ft").toPathsString();</symbol>
</scenario>

```

Fig.6.94. The scenario for the command to create a hierarchy of directory AGLO

Figure 6.94 shows the scenario for AGLO that aims to create a hierarchy of directories.

Two numbers are first randomly generated that represents the number of directories to be created and the level to which this hierarchy will be achieved. These numbers are required because this hierarchy is generated in the background as a tree. For this tree, the numbers represent the number of nodes and the height. The directory tree is generated. The name of each directory is chosen at random from an existing list of names to which is added a randomly chosen two-digit number.

In the variable *st* the directory tree is represented in indented level text format. In the variable *comenzi* is retained the answer consisting of the correct commands written each in a line. These commands represent all paths in the tree as a string.

Theory

The following hierarchy is considered:

```

alfa
*beta
**gama
***delta
***omega

```

The commands for achieving this hierarchy are:

```

mkdir -p alfa/beta/gama/delta
mkdir -p alfa/beta/gama/omega

```

Fig.6.95. The theory section for the command to create a hierarchy of directory using `-p` option AGLO

A static example is represented in the theory section, as can be seen in Figure 6.95. This example comprises an indented representation of a hierarchy of directories and then the corresponding lines of code for its realization. In this exercise, the student must know how to use the `-p` option correctly. For this, he also must know how to go through the folder tree in preorder.

Question

Using the `mkdir` command with the `-p` option in the current directory, create the directory hierarchy in the figure. An order will be written on each line.

```
Uniform15
*Tango21
**November40
**Hotel197
***X-ray79
```

Answers

```
mkdir -p Uniform15/Tango21/November40/
mkdir -p Uniform15/Tango21/Hotel197/X-ray79/
```

You got 10 out of 10! Evaluate

Feedbacks

Basically, you had to write a command for each path in the tree.

```
mkdir mkdir OK
-p -p OK
Uniform15/Tango21/November40/ Uniform15/Tango21/November40/ OK

mkdir mkdir OK
-p -p OK
Uniform15/Tango21/Hotel197/X-ray79/ Uniform15/Tango21/Hotel197/X-ray79/ OK
```

Fig.6.96. The question and answer sections for the command to create a hierarchy of directory using `-p` option AGLO

In the question section, the student receives a hierarchy of folders for which he must write specific commands using the `-p` option. It is also emphasized to the student that he must write only one command on each line of the answer. Figure 6.96

shows an implementation of the question, answer, and feedback sections in an exercise to create a folder hierarchy using the `-p` option.

The feedback section consists of two parts, a static one that tells the student that he had to write a command for each path in the tree. And a dynamic part, in which the student's answer is compared to the correct answer.

The following AGLO aims to create a directory hierarchy without using the `-p` option. In this case, each folder must be created individually. To do this the student must know how to go through the folder trees in preorder.

```

<scenario>
  <text>
A random tree is generated containing various names of number of nodes n.
It translates into indented text with the pre-marking.
It goes on a creation in preorder.
  <pre>
cd director
mkdir alfa
cd .. # return to parent director
  </pre>
  </text>
  <symbol name="n" type="integer">random(5,8,0);</symbol>
  <symbol name="ft" type="object">new FolderTree({"nNoOfNodes" : v("n")});
</symbol>
  <symbol name="st" type="string">v("ft").toHTML();</symbol>
  <symbol name="comenzi" type="string">v("ft").toMkdirCdPathsString();
</symbol>
</scenario>

```

Fig.6.97. The scenario for the command to create a hierarchy of directory without `-p` option
AGLO

Figure 6.97 shows the AGLO scenario that aims to create a folder hierarchy without using the `-p` option. In the first part, there is a text in which the idea of the exercise is explained in natural language. This is useful for the developer who wants to reuse this scenario.

Initially, the number of nodes of the tree is generated randomly, i.e. the number of folders that will exist in the respective hierarchy. The folder tree is then generated using the constructor function. The variable *st* will retain the indented representation of the tree as a string. The *comenzi* variable will retain the correct instructions needed to create the generated hierarchy.

In the theory section, the student is presented with a static example of a folder hierarchy as well as the commands needed to create it. What the student needs to remember is that a subfolder cannot be created without the `-p` option until we have made the parent folder the current folder. To return to a certain level or to the root folder he will use the `cd..` command.

Question

Create the directory hierarchy below without using the -p option, just using the commands:

```
mkdir nume
cd nume
cd ..
```

```
Uniform17
*Oscar55
*Quebec98
*Quebec19
**Mike29
**Charlie46
***India68
```

Answers

```
cd Quebec19
mkdir Mike29
mkdir Charlie46
cd Charlie46
mkdir India68
cd ..
```

You got 10 out of 10! [Evaluate](#)

Fig.6.98. The question and answer section for the command to create a hierarchy of directory without -p option AGLO

In the question section, the student is reminded of the commands he must use and is given a hierarchy for which he must write the correct commands, see Figure 6.98.

In this case, depending on the complexity of the hierarchy, the answer contains more command lines than when using the -p option.

```
Feedbacks

You had to create each folder using the mkdir command, and every time you entered a
subfolder you had to use cd .. to return to the parent.

mkdir mkdir OK
Uniform17 Uniform17 OK

cd cd OK
Uniform17 Uniform17 OK

mkdir mkdir OK
Oscar55 Oscar55 OK

mkdir mkdir OK
Quebec98 Quebec98 OK

mkdir mkdir OK
Quebec19 Quebec19 OK

cd cd OK
Quebec19 Quebec19 OK

mkdir mkdir OK
Mike29 Mike29 OK

mkdir mkdir OK
Charlie46 Charlie46 OK

cd cd OK
Charlie46 Charlie46 OK
```

Fig.6.99. The feedback section for the command to create a hierarchy of directory without `-p` option AGLO

In the feedback section, the student is reminded of what he had to do, namely that he had to create each folder using the *mkdir* command, and every time he entered a subfolder he had to return to the parent folder using the *cd..* command. In addition, below, the student's answer is compared to the correct answer on blocks of words.

6.3.2. Commands for files AGLOs

Regarding working with files, the first AGLO refers to creating a file. In this scenario, the student must know how to use the *touch* command correctly.

The theory part presents a static example in which the file with the alpha name is created.

In the question section, a file name is randomly generated. It is required to write the command to create the respective file.

The screenshot displays an AGLO (Assessment Generator for Learning Objectives) interface. It is divided into four main sections: Theory, Question, Answers, and Feedbacks. The Theory section contains the instruction 'The instruction used to create the file with the alpha name is: touch alpha'. The Question section asks the user to 'Using the touch command create the file with the name Papa74.mp3.'. The Answers section shows a text input field containing the command 'touch Papa74.mp3', followed by a score of 'You got 10 out of 10!' and an 'Evaluate' button. The Feedbacks section provides a static indication: 'Basically, you had to write a single command to create the file.' Below this, a code block shows the correct command and its output: 'touch touch OK' and 'Papa74.mp3 Papa74.mp3 OK'.

Theory

The instruction used to create the file with the alpha name is:
touch alpha

Question

Using the touch command create the file with the name Papa74.mp3.

Answers

```
touch Papa74.mp3
```

You got 10 out of 10!

Feedbacks

Basically, you had to write a single command to create the file.

```
touch touch OK
Papa74.mp3 Papa74.mp3 OK
```

Fig.6.100. An instance of the command to create a file AGLO

Figure 6.100 shows an instance of AGLO that aims to create a file named Papa74.mp3. We have given the correct answer in this example, namely the command touch Papa74.mp3.

In the feedback section, a static indication is provided, namely that a simple command had to be written. Then the answer given is compared with the correct answer.

The following AGLO targets the population of a folder tree with a given file.

To create this AGLO in the script section, a folder tree and a file name are randomly generated. In order to achieve this requirement, the student must go through the tree in pre-order.

In order to follow the essentials, namely only the correct application of the specific commands, we chose to populate them with the same file name.

The theory section shows how to create the alpha.txt file in the betta folder. What is wanted is to see that the *touch* command is applied.

Theory

The instruction used to create the file alpa.txt in the folder betta is:
touch betta/alpha.txt

Question

Using the touch command, create a file named Uniform68.doc in each of the folders in the tree shown in the following figure:

```

Mike24
 *Mike94
 *Hotel184
 *Quebec22
 **Romeo7
 ***Romeo60

```

Answers

```

touch Mike24/Uniform68.doc
touch Mike24/Mike94/Uniform68.doc
touch Mike24/Hotel184/Uniform68.doc
touch Mike24/Quebec22/Uniform68.doc
touch Mike24/Quebec22/Romeo7/Uniform68.doc
touch Mike24/Quebec22/Romeo7/Romeo60/Uniform68.doc

```

You got 10 out of 10!

[Evaluate](#)

Feedbacks

Basically, you had to write a single command to create the file, in which the file access path is used .

```

touch touch OK
Mike24/Uniform68.doc Mike24/Uniform68.doc OK

touch touch OK
Mike24/Mike94/Uniform68.doc Mike24/Mike94/Uniform68.doc OK

touch touch OK
Mike24/Hotel184/Uniform68.doc Mike24/Hotel184/Uniform68.doc OK

touch touch OK
Mike24/Quebec22/Uniform68.doc Mike24/Quebec22/Uniform68.doc OK

```

Fig.6.101. An instance of the command to populate a hierarchy of directory AGLO

In the question section, the student receives the file tree with the corresponding hierarchy and the name of the file with which this hierarchy must be populated. In addition, in the statement of the question, it is mentioned that it is necessary to use the touch command, as can be seen in Figure 6.101.

In this case, the student's answer includes several commands that contain the access ways to each folder in which the requested file is created.

In the feedback section is compared each command entered by the student with the correct answer so that in case of an error he knows where he did not write correctly.

The following AGLO also aims to popularize a hierarchy of folders with a file, only this time the folders do not exist but must also be created.

In this case, the student must know that in order to create the file in a certain folder, he has to follow the following steps:

- creating the folder;
- accessing the folder as the current folder;
- creating the file;
- returning to the parent folder.

When solving this AGLO, these steps must be applied to each folder in the hierarchy.

```
<scenario>
. . .
<symbol name="file" type="object">new File(1,null);</symbol>
<symbol name="fisier" type="string">v("file");</symbol>
<symbol name="n" type="integer">random(5,8,0);</symbol>
<symbol name="ft" type="FolderTree">new FolderTree({"noOfNodes" :
v("n")});</symbol>
<symbol name="st" type="string">v("ft").toString();</symbol>
<symbol
name="comenzi"
type="string">
v("ft").toPathFilesString(v("fisier"));</symbol>
</scenario>
```

Fig.6.102. The scenario of the command to create and populate a hierarchy of directory AGLO

As can be seen in Figure 6.102, the following variables are required to make this AGLO scenario:

- a *file* object, namely the file with which we will populate the folders, and its name as a string,
- a natural number that will represent the number of folders,
- a *FolderTree* object, namely the hierarchy of folders to be created and populated, and its representation as a string,
- a string that represents the correct answer.

In order to have an accessible exercise, we chose that the number of folders in the hierarchy should be between 5 and 8, and the nesting level up to which it can be 5.

Theory

The following hierarchy of folders is given:

```
A
*B
*C
```

The commands needed to populate these folders with a file named fisier are:

```
mkdir A
touch fisier
cd A
mkdir B
cd B
touch fisier
cd..
mkdir C
touch fisier
cd..
```

At the end it returns to the root folder.

Question

Using the touch command create the file with the name Uniform0.js in all folders in the tree shown in the following figure, returning to the end in the root folder.

```
Quebec60
*Oscar38
*Alpha31
**Charlie12
**Bravo43
```

Fig.6.103. The question and answer section for the command to create a hierarchy of directory without -p option AGLO

In the theory section, the student is given an example of a hierarchy with three folders and the commands needed to populate it with files. In this case, the student must know that each file is created with the touch command when the folder where it is to be created has been established as the current folder. After creating the file, return to the root folder.

The static example also has the role of reminding the student of the commands he must use:

- mkdir - to create a folder;
- cd - to make a folder the current folder;

- cd.. - to return to the root folder;
- touch - to create a file.

Answers

```
mkdir Charlie12
cd Charlie12
touch Uniform0.js
cd ..
mkdir Bravo43
cd Bravo43
```

You got 10 out of 10!

Evaluate

Feedbacks

You had to create each folder using the mkdir command and every time you went into a subfolder to create a file, you had to use cd .. to go back to the parent.

```
mkdir mkdir OK
Quebec60 Quebec60 OK

cd cd OK
Quebec60 Quebec60 OK

touch touch OK
Uniform0.js Uniform0.js OK

mkdir mkdir OK
Oscar38 Oscar38 OK

cd cd OK
Oscar38 Oscar38 OK

touch touch OK
Uniform0.js Uniform0.js OK
```

Fig.6.104. The question and answer section for the command to create a hierarchy of directory without -p option AGLO

In this case, the answer consists of several command lines.

In the feedback section, the student's answer is compared to the correct answer. Figure 6.104 shows a correct response to an instantiation of this AGLO.

The following AGLO targets command specific to changing the access permissions of a file.

For this, a file name and three natural numbers between 0 and 7 are randomly generated. These numbers will represent the octal digits that show the access permissions of the three user classes to the desired file.

Theory

Modifying the access permission of the three user classes with the help of the chmod command and the octal numbers:

- 0 --- no permission
- 1 --x execute
- 2 -w- write
- 3 -wx write and execute
- 4 r-- read
- 5 r-x read and execute
- 6 rw- read and write
- 7 rwx read, write and execute

Question

Using the octal digit chmod command set the following permissions on the file Delta95.txt:

```
for the user read,  
for the group no permission and  
for others no permission
```

Answers

```
chmod 400 Delta95.txt
```

You got 10 out of 10!

[Evaluate](#)

Feedbacks

Basically, you had to write a single command.

```
chmod chmod OK  
400 400 OK  
Delta95.txt Delta95.txt OK
```

Fig.6.105. An instantiation of setting the file access permissions AGLO

In the theory section, the student receives, in the form of a static text, the meaning of the access permissions to a file, as can be seen in Figure 6.105.

In the question, the student can see for each group of users which access permissions must be set. In the example from Figure 6.105, the student must set the following permissions for Delta95.txt file: for the user read and for the group and others no permission.

The answer is the correct application of the *chmod* command, namely `chmod 400 Delt95.txt`.

The following AGLO aims to upload a new file system to the main file system of the system.

When instantiating this learning object, a file name and a system file are randomly generated.

Theory

The mount command loads a new file system into the system's main file system.

Question

Using the mount command to create the directory with the name Golf39.java in the system file ext2 /dev/sda6.

Answers

```
mount -t Golf39.java /ext2 /dev/sda6
```

 Evaluate

You got 10 out of 10!

Feedbacks

Basically, you had to write a single command.

```
mount mount OK
-t -t OK
Golf39.java Golf39.java OK
/ext2 /ext2 OK
/dev/sda6 /dev/sda6 OK
```

Fig.6.106. An instantiation of the mount command AGLO

The theory section presents the action of the mount command.

In the question section, the student is instructed to use the mount command to create the given file Golf39.java in the received system file ext2/dev/sda6.

The answer is the correct application of the mount command.

In the feedback section, the student's answer is compared with the correct answer and is reminded that he had to write a single command.

6.3.3. Commands for processes AGLOs

Next, we made some AGLOs aimed at working with processes: displaying processes, moving a process from, or to the background/foreground.

The first application aims to display background processes.

We want to start in the background a process that displays all files that have a certain extension and start with a certain set of characters. To do this, a character set for the file names and an extension are randomly generated.

The screenshot displays three sections of an AGLO interface:

- Question:** A pink header with the text "Question". Below it, a white box contains the instruction: "Start a background process that displays all those files the name of which starts with beta and end with c."
- Answers:** A blue header with the text "Answers". Below it, a text input field contains the command `ls beta*.c &`. Below the input field, it says "You got 10 out of 10!". To the right of the input field is a blue button labeled "Evaluate".
- Feedbacks:** A yellow header with the text "Feedbacks". Below it, a white box contains the text: "Basically, you had to write a single command to start in background the ls command." Below this text is a grey box containing a terminal-like output:

```
ls ls OK
beta*.c beta*.c OK
& & OK
```

Fig.6.107. An instantiation of the command to start a display process in the background AGLO

In the theory section, the student is reminded that the `ls` command is used to display all files with certain properties.

Figure 6.107 shows an instance of this AGLO in which the student is asked to start in the background a process that displays all the files whose names start with *beta* and have the extension *c*.

The answer consists of a single line of code that represents the display command followed by the & sign, which represents the start in the background.

In the feedback section, the correct answer and the student's answer are compared, so you can see where the student is wrong if necessary.

The next AGLO aims to complete a process without sending it in the background. For this, a process number with four digits is randomly generated.

Theory

Using the specific command kill a process.

A list of processes looks like this:

USER	PID	START	TIME	COMMAND
root	1	15:30	0:04	init
root	2	15:30	0:00	[keventd]
root	3	15:30	0:00	[ksoftirqd_CPU0]
root	3456	17:05	0:05	[cpuhp]
root	1283	17:30	0:03	[kthreadd]
keeper	719	16:27	0:01	gedit
keeper	741	16:28	0:01	pine
keeper	796	16:53	0:05	rogue
keeper	825	17:04	0:00	grep rogue

Question

Using the kill command you finish the 1323 process.

Fig.6.108. The theory and question sections for the command to kill a process AGLO

In the theory section, the student is reminded that a process ends using the kill command. He is also given a realistic example of a list of nine processes that could run on a computer. Figure 6.108 shows the static part that includes the theory from this exercise.

In the question section, the student is asked to stop the process with the indicated PID, i.e. the previously randomly generated number.

The answer lies in the correct application of the kill command, namely using the -9 option.

The image shows two sections of a learning interface. The top section, titled 'Answers', has a light blue header. Below it is a text input field containing the command 'kill -9 1323'. To the right of the input field is a blue button with a white eye icon and the text 'Evaluate'. Below the input field, the text 'You got 10 out of 10!' is displayed. The bottom section, titled 'Feedbacks', has a light yellow header. Below it, the text 'To kill a process, simply type' is followed by 'kill -9 PID'. A grey box contains a comparison of the student's answer with the correct syntax: 'kill kill OK', '-9 -9 OK', and '1323 1323 OK'.

Fig.6.109. The answer and feedback sections for the command to kill a process AGLO

The feedback section consists of two parts:

- the first consists of a static text in which the correct syntax of the kill command is presented,
- the second dynamic in which the student's answer is compared with the expected correct answer.

The next AGLO aims to complete a background process.

To achieve the scenario targeted by this AGLO, a realistic list is generated consisting of three processes that are supposed to run in the background. The numbers representing the PID of the processes are randomly generated.

Also, a number between 1 and 3 is randomly generated, which will represent which process in the list needs to be completed.

In the theory section, the student is reminded that he must use the kill command.

In the question section, besides the statement of the question, the student also receives the list of processes from which the one to be stopped is chosen.

The answer consists of a single command line that should contain the correct application of the specific kill command.

In the feedback section, the student receives both the correct syntax that had to be used for the kill command and the comparison of his answer with the correct answer.

Theory

Using the specific command kill a process.

Question

```
# jobs
[ 295]  Running          bash download-file.sh &
[1711]- Running          evolution &
[4380]  Running          myscrip &
[   4]+ Done             nautilus .
```

Using the kill command, finish the first process from background.

Answers

```
kill %295
```

[Evaluate](#)

You got 10 out of 10!

Feedbacks

To kill a process in the background, simply type
kill %PID

```
kill kill OK
%295 %295 OK
```

Fig.6.110. An instantiation of the command to kill a process in the background AGLO

The following AGLO aims to bring a process from background to foreground. To achieve the scenario covered by this AGLO, a realistic list is generated consisting of three processes that should run in the background. The numbers that represent the process PID are randomly generated. It is also randomly chosen which of the three processes will be moved to the foreground.

The screenshot shows a three-part interface for a quiz question. The top part, titled 'Question', contains the text 'The following processes run in the background:' followed by a table of three running processes. The middle part, titled 'Answers', features a text input field containing the command 'fg %2457' and an 'Evaluate' button. The bottom part, titled 'Feedbacks', provides the correct command 'fg %PID' and a comparison of the student's answer with the correct one.

Question

The following processes run in the background:

[248]	Running	bash download-file.sh &
[1062]	Running	evolution &
[2457]	Running	myscript &

Using the specific command bring in foreground the third process.

Answers

fg %2457

You got 10 out of 10!

Feedbacks

To bring a process to the foreground, use the following command:
fg %PID

```
fg fg OK
%2457 %2457 OK
```

Fig.6.111. An instantiation of the command to move a process from background to foreground
AGLO

In the question section, besides the statement of the question, the student also receives the list of processes from which the one to be moved is chosen see Figure 6.111.

The answer consists of a single command line that should contain the correct application of the specific *fg* command.

In the feedback section, the student receives both the correct syntax that had to be used for the *fg* command and the comparison of his answer with the correct answer.

6.3.4. Commands for disk partitioning AGLOs

The next two AGLOs focus on using the *fdisk* command.

In the first ALGO, the use of *fdisk* command options is followed.

The screenshot displays a learning interface with four distinct sections:

- Theory:** A green header with the text "How to use fdisk command."
- Question:** A red header with a text box containing the question: "After the command: `fdisk /dev/sda1`, what will you type to print the partition table."
- Answers:** A blue header with a text input field containing the letter 'p'. Below the field, it says "You got 10 out of 10!" and there is an "Evaluate" button.
- Feedbacks:** A yellow header with the text "Basically, you had to write a single letter corresponding to the option." Below this, a text box shows the user's input "p p OK".

Fig.6.112. An instantiation of using the options of the fdisk command AGLO

The student should know that, depending on the option he chooses, he can have the following options:

- accessing a bootable flag;
- editing the label of a disc;
- delete a partition;
- enumeration of known partitions;
- print the help menu;
- adding a new partition;
- print the partition table.

In the theory section, the first statement is a realistic use of the fdisk command on a randomly generated partition. Following this command, the programmer has to choose one of the seven options listed above. The student receives one of them at random and must say which is the letter corresponding to that option.

The letter corresponding to the option to be typed represents the correct answer to the question in the AGLO.

In the feedback section, the student is reminded that he had to choose only one letter that represents the required option.

The following AGLO aims to view a partition of the hard disk.

The screenshot displays a learning management system interface for an AGLO. It is divided into four colored sections: a green 'Theory' section, a red 'Question' section, a blue 'Answers' section, and a yellow 'Feedbacks' section. The 'Theory' section contains the text 'The fdisk command is used to view the partition of a hard disk.' The 'Question' section asks the user to 'View the hard disk partition /dev/sdc.' The 'Answers' section features a text input field containing the command 'fdisk -l /dev/sdc', a 'You got 10 out of 10!' message, and an 'Evaluate' button. The 'Feedbacks' section provides a message: 'Basically, you had to type the command null to view the partition.' Below this message is a code block showing the correct command sequence: 'fdisk fdisk OK', '-l -l OK', and '/dev/sdc /dev/sdc OK'.

Theory

The fdisk command is used to view the partition of a hard disk.

Question

View the hard disk partition /dev/sdc.

Answers

```
fdisk -l /dev/sdc
```

You got 10 out of 10!

Evaluate

Feedbacks

Basically, you had to type the command null to view the partition.

```
fdisk fdisk OK
-l -l OK
/dev/sdc /dev/sdc OK
```

Fig.6.113. An instantiation of the command to view a partition AGLO

As can be seen in Figure 6.113 the student receives a randomly generated hard drive name.

The answer is represented by the correct use of the fdisk command together with the -l option.

In the feedback section, the student receives the correct command to be able to correct any mistakes.

6.3.5. Package management AGLOs

The following AGLOs aim the package management. In Linux to install or uninstall a software package depending on the version of the operating system, two variants are included in the next two AGLOs.

The first AGLO aims to install or uninstall a program under Linux versions such as Debian, Ubuntu, or Knoppix.

The screenshot displays an AGLO interface with four main sections:

- Theory:** A green header section containing the text "Using the specific tool install or uninstall a program on Linux (Dabian,Ubuntu,Knoppix)."
- Question:** A red header section containing the text "Install program1.deb."
- Answers:** A blue header section containing a text input field with the command `apt-get install $program1.deb`, an "Evaluate" button, and the feedback "You got 10 out of 10!".
- Feedbacks:** A yellow header section containing a paragraph of text and a code block showing the command execution output.

The feedback text states: "There are two main tools around APT: apt-get and apt-cache. apt-get is for installing, upgrading, and cleaning packages, while apt-cache is used for finding new packages."

```
apt-get apt-get OK
install install OK
$program1.deb $program1.deb OK
```

Fig.6.114. An instantiation of the apt command AGLO

In the theory part, it is explained to the student that he will have to install or uninstall a program in a certain version of Linux, as can be seen in Figure 6.114.

The option to install or uninstall is randomly generated so that in one instance the student can have only one of them to perform. The program name is also randomly generated from a list of four possibilities.

The answer consists of the correct use of the specific command to install or uninstall the received program.

In the feedback section, the student is reminded of the theoretical part, namely that there are two main tools around APT:

- apt-get is for installing, upgrading, and cleaning packages,
- apt-cache is used for finding new packages

The next AGLO aims to install or uninstall a program under Linux versions such as Fedora, Red Hat or Mandriva.

The screenshot displays a learning management system interface with four distinct sections:

- Theory:** A green header section containing the text: "Using the specific tool install or uninstall a program on Linux (Fedora,Red Hat, Mandriva)." Below this is a white text area.
- Question:** A red header section containing the text: "Install program3.rpm." Below this is a white text area.
- Answers:** A blue header section containing a text input field with the command `yum install $program3.rpm`. Below the input field is a blue button labeled "Evaluate". Below the button, the text "You got 10 out of 10!" is displayed.
- Feedbacks:** A yellow header section containing the text: "YUM is a free and open-source command-line package-management utility for computers running the Linux operating system using the RPM Package Manager. Basically, you had to write a single command to install or uninstall the program." Below this text is a grey code block containing the following lines:

```
yum yum OK
install install OK
$program3.rpm $program3.rpm OK
```

Fig.6.115. An instantiation of the yum command AGLO

In the theory part, it is explained to the student that he will have to install or uninstall a program in a certain version of Linux, as can be seen in Figure 6.115.

The option to install or uninstall is randomly generated so that in one instance the student can have only one of them to perform. The program name is also randomly generated from a list of four possibilities.

The answer consists of the correct use of the specific command to install or uninstall the received program.

In the feedback section, the student is reminded of the theoretical part, namely that YUM is a free and open-source command-line package-management utility for computers running the Linux operating system using the RPM Package Manager.

6.3.6. Administrative commands AGLOs

The following applications target administrative commands.

The first AGLO in this category aims to add a user. To create this scenario, a username is randomly generated.

The screenshot displays an AGLO interface with four main sections:

- Theory:** A green header with the text "Adding a user using the command: useradd."
- Question:** A red header with the text "Using the useradd command add the user Juliet42."
- Answers:** A blue header with a text input field containing the command `useradd -s /bin/bash -m -d/home/Juliet42`. Below the input field, it says "You got 10 out of 10!" and there is a blue "Evaluate" button.
- Feedbacks:** A yellow header with the text "Basically, you had to write a single command to add the user." Below this, a grey box shows a comparison of the student's answer to the correct one:

```
useradd useradd OK
-s -s OK
/bin/bash /bin/bash OK
-m -m OK
-d/home/Juliet42 -d/home/Juliet42 OK
```

Fig.6.116. An instantiation of the command to add a user AGLO

In the theory section, he is reminded that he wants to use the *useradd* command, see Figure 6.116.

The question requires adding the user with the random name generated.

The answer consists of the correct use of the *useradd* command.

In the feedback section, the student's answer is compared with the correct one, specifying that he had to write a single command.

The next AGLO aims to add a user to a certain group. To create this scenario, a username is randomly generated and a group is randomly chosen.

The screenshot displays a learning management system interface with four distinct sections:

- Theory:** A green header section containing the text "Adding a user using the command: usermod."
- Question:** A red header section containing the text "Using the usermod command add the user Bravo95 in the group guests."
- Answers:** A blue header section containing a text input field with the command `usermod -aG guests Bravo95` and an "Evaluate" button. Below the input field, it states "You got 10 out of 10!".
- Feedbacks:** A yellow header section containing the text "Basically, you had to write a single command to add the user to the specified group." and a code block showing the output of the command:

```
usermod usermod OK
-aG -aG OK
guests guests OK
Bravo95 Bravo95 OK
```

Fig.6.117. An instantiation of the command to add a user to a group AGLO

In the theory section, he is reminded that he wants to use the *usermod* command.

The question asks to add the user with the random name generated in one of the groups: administrator, developers, designers, guests.

The answer consists of the correct use of the *usermod* command.

In the feedback section, the student's answer is compared with the correct one, specifying that he had to write a single command.

The next AGLO in this category aims to generate specific information about a user. To create this scenario, a username is randomly generated.

The screenshot displays a learning interface with four distinct sections:

- Theory:** A green header section containing the text: "Using the specific command, generate information about a user, namely the login name, name, directory, shell, login time, email, and plan of the user."
- Question:** A red header section containing the text: "Using the specific command generate specific information about the user with the name Kilo91."
- Answers:** A blue header section containing a text input field with the command `finger Kilo91`, a blue "Evaluate" button, and the text "You got 10 out of 10!".
- Feedbacks:** A yellow header section containing the text: "The finger command displays the login name, name, directory, shell, login time, email, and plan of the user." Below this is a grey box showing the command output: `finger finger OK` and `Kilo91 Kilo91 OK`.

Fig.6.118. An instantiation of the command to generate information about a user AGLO

In the theory section, he is reminded that he wants to generate specific information about a particular user, see Figure 6.118.

The question requires the display of specific information about the user with the random name generated. In the above instantiation the student is asked to write the command that display specific information about Kilo91 user.

The answer consists of the correct use of the finger command, namely finger Kilo91.

In the feedback section, the student's answer is compared with the correct one. The student is also reminded that the finger command displays the login name, name, directory, shell, login time, email, and plan of the user.

The next AGLO aims to change the password of a user. To create this scenario, a username is randomly generated.

The screenshot displays a learning management system interface with four distinct sections:

- Theory:** A green header section containing the text: "Changing a user's password is done using the specific command."
- Question:** A red header section containing the text: "Using the specific command change the user's password with the name Papa56."
- Answers:** A blue header section containing a text input field with the command `passwd Papa56`. Below the field is a blue "Evaluate" button and the text "You got 10 out of 10!".
- Feedbacks:** A yellow header section containing the text: "Basically, you had to type a single command to change the user's password." Below this is a grey box showing a comparison of the student's answer with the correct one:

```
passwd passwd OK
Papa56 Papa56 OK
```

Fig.6.119. An instantiation of the command to change the password of a user AGLO

In the theory section, he is reminded that he wants to change the password of a certain user, see Figure 6.119.

In the question section, the student is asked to change the password of the user with the randomly generated name, namely Papa56.

The answer consists of the correct use of the `passwd` command, thus in the answer section the student has to complete a single command.

The feedback section compares the student's answer with the correct one, namely `passwd Papa56`.

The next AGLO aims to delete a user. To create this scenario, a username is randomly generated.

The screenshot displays a learning management system interface with four main sections:

- Theory:** A green header bar with the text "Theory". Below it, a statement reads: "A specific command is used to delete a user."
- Question:** A red header bar with the text "Question". Below it, the question text reads: "Using the specific command delete the user by name Delta38."
- Answers:** A blue header bar with the text "Answers". Below it, a text input field contains the command: `userdel -r Delta38`. To the right of the input field is a blue button labeled "Evaluate". Below the input field, the text "You got 10 out of 10!" is displayed.
- Feedbacks:** A yellow header bar with the text "Feedbacks". Below it, the feedback text reads: "Basically, you had to write a single command to delete the user." Below this text is a grey box containing a comparison of the student's answer with the correct one:

```
userdel userdel OK
-r -r OK
Delta38 Delta38 OK
```

Fig.6.120. An instantiation of the command to delete a user AGLO

In the theory section, the student is reminded that he wants to delete a certain user.

The question asks the student to delete the user with the random name generated, namely Delta38.

The answer consists of the correct use of the `userdel` command, namely `userdel Delta38`.

The feedback section compares the student's answer with the correct one, reminding that only one command had to be written.

6.3.7. Basic networking commands AGLOs

The following AGLOs target network-specific commands: view, configure or delete an address.

The first AGLO aims to manage and configuring network interfaces with the `ifconfig` command. To create this scenario, an option is randomly generated.

To learn how to use the `ifconfig` command, one of the variants can be generated:

- display all the active interfaces details;
- display information of all active or inactive network interfaces on server;
- display details about the eth0 specific network interface;
- activates a network interface with interface name eth0;
- deactivates the specified network interface with interface name eth0;
- assign the IP address 172.16.25.125 to the interface named eth0;
- set the network mask 255.255.255.224 to a given interface eth0.

Theory

Managing and configuring network interfaces with the ifconfig command.

Question

Activates an network interface with interface name eth0.

Answers

```
ifconfig eth0 up
```

You got 10 out of 10!

Evaluate

Feedbacks

Basically you had to use the ifconfig command with different options.

```
ifconfig ifconfig OK
eth0 eth0 OK
up up OK
```

Fig.6. 121. An instantiation of the ifconfig command AGLO

The question asks the student to solve one of the above options. As can be seen in Figure 6.121 is asked to activates a network interface with name eth0.

In the answer section, it needed to be written the correct uses of the ifconfig command, namely ifconfig eth0 up.

The feedback section compares the student's answer with the correct one, reminding that he has to use the ifconfig command with different options.

The next AGLO is about deleting an address.

The screenshot displays an AGLO interface with four sections: Theory, Question, Answers, and Feedbacks. The Theory section is green and contains the text 'Delete an address using the command: arp.' The Question section is red and contains 'Delete the 225.124.61.36 address.' The Answers section is blue and contains a text input field with 'arp -d 225.124.61.36', a 'You got 10 out of 10!' message, and an 'Evaluate' button. The Feedbacks section is yellow and contains the text 'Basically, you had to write a single command to delete the address.' followed by a code block showing the correct command and its output.

Theory

Delete an address using the command: arp.

Question

Delete the 225.124.61.36 address.

Answers

```
arp -d 225.124.61.36
```

You got 10 out of 10!

Feedbacks

Basically, you had to write a single command to delete the address.

```
arp arp OK
-d -d OK
225.124.61.36 225.124.61.36 OK
```

Fig.6.122. An instantiation of the command to delete an address AGLO

In the theory section, the student is reminded that he wants to use the arp command.

The question asks the student to delete a randomly generated address. In Figure 6.122 is captured an instantiation in which is asked to delete the 225.124.61.36 address.

The answer consists of the correct use of the arp command, namely arp -d followed by the address.

The feedback section compares the student's answer with the correct one, reminding that only one command had to be written.

The next AGLO aims to configure a new IP with netmask using the command: ifconfig. To accomplish this scenario, an IP address and a network interface are randomly generated.

The screenshot displays a multi-section interface for a learning object. It is divided into four horizontal panels: 'Theory' (green header), 'Question' (red header), 'Answers' (blue header), and 'Feedbacks' (yellow header). The 'Theory' panel contains the instruction: 'Configure a new IP with netmask using the command: ifconfig.' The 'Question' panel asks: 'Configure 167.115.67.28 a new IP address on eth3 with the mask 255.0.0.0 .'. The 'Answers' panel features a text input field containing the command: 'ifconfig eth3 167.115.67.28 netmask 255.0.0.0', a score indicator 'You got 8.33 out of 10!', and an 'Evaluate' button. The 'Feedbacks' panel provides a summary: 'Basically, you had to write a single command to configure the address.' followed by a terminal-style output showing the command being broken down into four sequential steps, each marked as 'OK': 'ifconfig ifconfig OK', 'eth3 eth3 OK', '167.115.67.28 167.115.67.28 OK', and 'netmask netmask OK', '255.0.0.0 255.0.0.0 OK'.

Fig.6.123. An instantiation of the command to configure an address AGLO

In the question section, the student is asked to configure a randomly generated address with a new IP address on a certain interface and with the mask 255.0.0.0.

The answer consists of the correct use of the ifconfig command.

In figure 6.115 we represented an instantiation with the wrong answer of this AGLO. As you can see, the student's answer is compared to the correct one sequentially on the blocks so that he knows where he went wrong. Also, the grade received is proportional to how correct the answer is.

The following AGLO aims to manually enable or disable a network interface without using the ifconfig command.

The screenshot displays a four-part AGLO interface. The 'Theory' section (green header) contains the text: 'A network interface can be activated or deactivated manually without using ifconfig command.' The 'Question' section (red header) contains: 'Activated manually eth3 network interface.' The 'Answers' section (blue header) features a text input field with 'ifup eth3', a 'You got 10 out of 10!' message, and an 'Evaluate' button. The 'Feedbacks' section (yellow header) contains: 'The ifdown command take a network interface down and the ifup command bring a network interface up.' Below this is a code block with the text: 'ifup ifup OK' and 'eth3 eth3 OK'.

Fig.6.124. An instantiation of the command to activated/deactivated an interface AGLO

In the theory section, the student has presented what the exercise aims at, namely the manual activation or deactivation of a network interface.

The question section required activating or deactivating depending on the randomly generated option of a network interface that is also randomly generated.

The feedback section consists of two parts:

- a static one in which the student is reminded that the ifdown command take a network interface down and the ifup command bring a network interface up;
- a dynamic in which the student's answer is compared which the correct expected answer.

The following AGLO aims to Adding (Deleting) a route via a gateway connected via a network interface.

Theory

Add or delete a route via a gateway connected via a network interface using the command: route.

Question

Add a route via 104.196.102.43 gateway connected via eth0 network interface.

Answers

```
route add default gw 104.196.102.43 eth0
```

You got 10 out of 10!

Feedbacks

Basically, you had to write a single command.

```
route route OK
add add OK
default default OK
gw gw OK
104.196.102.43 104.196.102.43 OK
eth0 eth0 OK
```

Fig.6.125. An instantiation of the route command AGLO

In the theory section, the student is presented with what the exercise aims at, namely the addition or deletion of a route via a gateway connected via a network interface using the command: route.

The question section required deleting or adding depending on the randomly generated option of a route. In Figure 6.125 is represented an instantiation of the AGLO in which the student is asked to add a route via 104.196.102.43 gateway connected via eth0 network interface.

The answer consists of the correct use of the route command, namely route add default gw 104.196.102.43 eth0.

The feedback section provides the student with information about what he or she should write and provides a comparison of his or her answer with the correct one.

6.4. Summary

For the Data Structures and Algorithms discipline, we grouped the exercises into seventeen topics. The topics covered are:

- several searches algorithms: the linear search, the linear search with sentinel, the binary search, the interpolation search;
- several sorting algorithms: the insert sort, the select sort, the bubble sort, the shell sort, the heap sort, the quick sort, the inter-classification with three bands, the inter-classification with four bands, the natural inter-classification;
- the simple linked linear lists;
- the double linked linear lists;
- the stacks;
- the queues.

The distribution of the AGLOs by topics is represented in Table 6.1.

No.	The topics	Number of AGLOs
1.	Linear search	13
2.	Linear search with sentinel	5
3.	Binary search	6
4.	Interpolation search	6
5.	Insert search	8
6.	Select search	5
7.	Bubble sort	3
8.	Shell sort	3
9.	Heap sort	3
10.	Quick sort	5
11.	Inter-classification with three bands	1
12.	Inter-classification with four bands	1
13.	The natural inter-classification	1
14.	Simple linked linear lists	20
15.	Double linked linear lists	11
16.	Stacks	4
17.	Queues	2

Table 6. 1. Distribution of AGLOs on topics for DSA discipline

For the discipline of algorithms analysis and design, we have made twelve AGLOs aimed at notions related to the notion of tree, and five AGLOs related to the notion of graph.

For the Operating Systems discipline, we grouped the AGLOs by topics as follows:

- the commands for working with directors;
- the commands for working with files;
- the process commands;
- the commands for disk partitioning;
- the commands for package management;
- the administrative commands;
- the commands for networks.

The distribution of the AGLOs by topics is represented in Table 6.2.

No.	The topics	Number of AGLOs
1.	Commands for working with directors	3
2.	Commands for working with files	5
3.	Process commands	5
4.	Commands for disks	2
5.	Commands for package management	2
6.	Administrative commands	5
7.	Commands for networks	5

Table 6.2. Distribution of AGLOs on topics for OS discipline

We can conclude that we have made 141 AGLOs targeting different ITC disciplines.

7. PROTOTYPE IMPLEMENTATION

7.1. The DSEL platform

In order to be able to use these AGLOs in the teaching-learning-evaluation activity, we considered it useful for them to be organized in a web page. Thus we created dsel.upt.ro. The platform is called DSEL because it was originally built to use AGLO in Data Structures discipline. This is a web page where the student can log in with a Facebook, Google or Yahoo existing account.

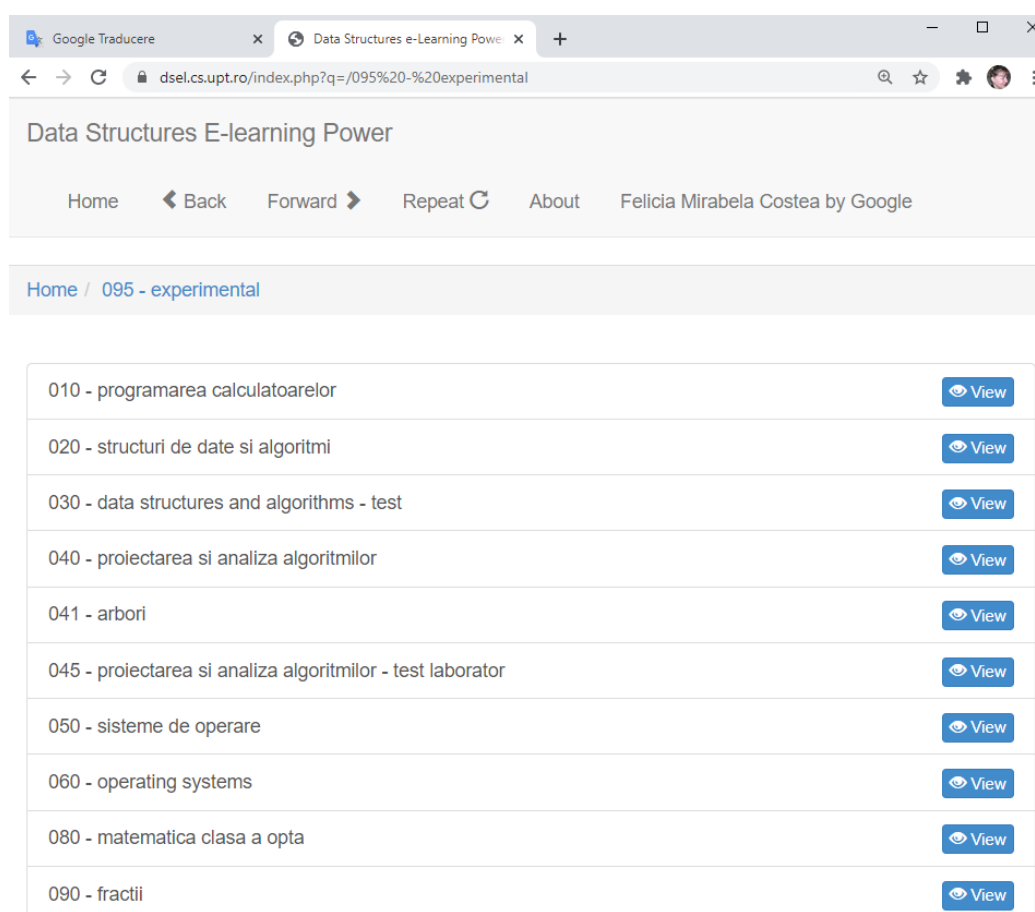


Fig.7.1. DSEL platform

The DSEL platform is built from an authentication module and an interface module. The interface module allows the student to navigate the LO tree. XML files containing AGLOs are stored in a hierarchy of directories arranged by discipline (see Figure 7.1). By activating the View button, the theme-specific AGLOs are accessed.

Each topic has a different AGLO number depending on the learning objectives. The grouping of the AGLOs is performed as follows: computer programming, data structures and algorithms, design and analysis of algorithms, trees, operating systems, mathematics for the eighth grade and, fractions. When the View button is activated, the student has access to the objects that target the respective topic, some of these topics are grouped on topics, and others are not.

The screenshot shows a web browser window with the URL `dsel.cs.upt.ro/index.php?q=/095%20-%20experimental/080%20-%20matematica%20clasa%20a%20opta/act050%20-%20inecuatie...`. The page header includes navigation links: Home, Back, Forward, Repeat, About, and Mirabela Costea by Google. The breadcrumb trail is: Home / 095 - experimental / 080 - matematica clasa a opta / act050 - inecuatie cu modul.xml.

The main content is divided into four sections:

- Theory:** A green header section containing the text: "A rezolva o inecuatie inseamna a ii determina multimea solutiilor." followed by the inequality $|2x + 5| \leq 7$ and its solution $[-6 ; 1]$.
- Question:** A pink header section with the text: "Rezolvati urmatoarea inecuatie:" followed by the inequality $|4x + 21| \leq 17$.
- Answers:** A blue header section with a text input field containing $[-9.5 ; -1]$. Below the input field, it says "You got 10 out of 10!" and there is an "Evalueaza" button.
- Feedbacks:** A yellow header section showing the feedback for the answer: $[-9.5 ; -1]$. Below it, the system response is: $[-9.5 [-9.5 OK ; ; OK -1] -1] OK$.

Fig.7.2. An AGLO Instantiation on DSEL Platform

The instantiation of an AGLO is represented in Figure 7.2. The AGLO sections visible to the student are the theory section, the question section, the answer section, and the feedback section. The feedback section is visible when the student requests the evaluation of the answer by activating the Evaluate button. When activating the button, the student's grade is automatically generated. If the student wants to repeat this type of exercise, it is enough to choose the Repeat option from the header menu. This allows them to access the same type of exercise, but the instantiation will be done with other randomly generated numbers.

The header of the page contains the Back and Forward options. They have the effect of accessing web pages like similar options in the browser. The Home label

takes the student to the home page of the platform from where he can chose another discipline.

lastName	sessionName	email	loginMethod	agloName	agloPath	tableOfSymbols	question	studentAnswer	computedAnswer	feedback
ROBERTA	Sierra8975	istocroberta@nbolcas.ro	Google	act030 - diferența de intervale.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=-7 b=4 c=-16 d=3 f1=[-7; 4] f2=[-16; 3] interv...	Diferența intervalelor [-7; -16] 3...	[-7; -16]	(3; 4]	(3; 4] -> OK [-7; -16] -> NOT OK
DARIUS	X-ray5767	istocdarius@nbolcas.ro	Google	act03 - amplificarea unei fractii ordinare.xml	/090 - fractii/act03 - amplificarea unei fractii o...	nr=20 num=12 amplific=2 f(object Object) raspuns=...	Amplificati 20/12 cu numarul 2.	40/4	40/24	Trebuia sa inmultesti atat numarul cat si numi...
ALEXANDRU	Charlie7553	gaialexandru@nbolcas.ro	Google	act080 - restrangeri pe baza formulelor.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=9 val=-; - f=0 intrebare=x^2 + 18*x + 81 raspuns=...	Restrangeti dupa formula : x^2 + 18*x + 81 = (x + 9)^2 (x + 81)^2 x^2 + 18*x + 81			(x + 9)^2 -> OK (x + 81)^2 -> NOT OK 18*x 9^2 -> OK
RIANA	Quebec7931	ignariana@nbolcas.ro	Google	act050 - inecuatie cu modul.xml	/095 - experimental/080 - matematica - clasa a opta/...	val=2,4,5,8,10 j=3 a=10 b=23 c=40 f=[-6,3; 1,7] r...	Rezolvati urmatoarea inecuatie -> [10*x + 23] ...	[39; 1]	[-6,3; 1,7]	[-6,3; 1,7] -> OK [1,7; ...] -> NOT OK
ALEXIA	Kilio3347	jurcanalexia@nbolcas.ro	Google	act085 - restrangeri pe baza formulelor.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=8 intrebare=x^2 - 64 raspuns=(x - 8)(x + 8)	Restrangeti dupa formula : x^2 - 64	(x - 8)(x + 8)	(x - 8)(x + 8)	(x - 8)(x + 8) -> OK (x - 8)(x + 8) -> OK (x - 8)(x + 8) -> OK
DARIUS	X-ray5767	istocdarius@nbolcas.ro	Google	act04 - simplificarea unei fractii ordinare.xml	/090 - fractii/act04 - simplificarea unei fractii o...	nr=13 num=11 f(object Object) raspuns=13/11	Simplificati fractia 13/11 astfel incat sa obtineti...	4/13	13/11	-> Fractia trebuia adusa la forma sa ireductib...
STEFANIA	Kilio52	borosstefania@nbolcas.ro	Google	act050 - inecuatie cu modul.xml	/095 - experimental/080 - matematica - clasa a opta/...	val=2,4,5,8,10 j=3 a=8 b=14 c=42 f=[-7; 3,5] rasp...	Rezolvati urmatoarea inecuatie -> [8*x + 14] <= ...	Multiplu de solutii: [1, 2, 3]	[-7; 3,5]	[-7; 3,5] -> OK Multiplu de solutii -> NOT OK
ALEXANDRU	Charlie7553	gaialexandru@nbolcas.ro	Google	act075 - formule de calcul prescurtat.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=12 raspuns=x^2 - 144	Calculati (x - 12)(x + 12) = x^2 - 144	(x - 12)(x + 12)	x^2 - 144	x^2 - 144 -> OK x^2 - 144 -> NOT OK x^2 - 144 -> NOT OK
AURA	Victor3966	costeaaura@nbolcas.ro	Google	act065 - inecuatii.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=2 b=14 c=48 f=[-31; +infnit] raspuns=[-31; +i...	Rezolvati urmatoarea inecuatie -> [2*x - 14] <= 4...	x apartine [-infnit; 17]	[-31; +infnit]	[-31; +infnit] -> OK [-31; +infnit] -> NOT OK
ROBERTA	Sierra8975	istocroberta@nbolcas.ro	Google	act010 - intersecții de intervale marginite.xml	/095 - experimental/080 - matematica - clasa a opta/...	a=3 b=20 c=17 d=-9 f1=[3; 20] f2=[-17; -9]	[3; 20] ∩ [-17; -9]	[3; 20] ∪ [-17; -9]	multimea vida	Intersecția intervalelor A și B este intervalul mu...

Fig.7.3. The LO repository with the students answers

The answers given by the students are stored in a LOR (see Figure 7.3), from where they can be accessed. Each record in this database has the following fields:

- id - is the serial number of the registration in the database;
- name - represents the student's name;
- firstName - the student first name;
- lastName - the student last name;
- sessionName - each login session receives a unique name;
- email - the email with which the login session was performed;
- loginMethod - the way in which the login was made, i.e. directly through the g-mail address or through a Facebook existing account;
- agloName - is the name of the AGLO that the student accessed;
- agloPath - is the path to the AGLO that the student accessed;
- tableOfSymbols - includes all the symbols from the respective AGLO scenario;
- question - the statement of the question;
- studentAnswer - the answer given by the student;

- computeAnswer - the correct answer calculated automatically;
- feedback - the feedback received at the evaluation by the student;
- noOfCorrectItems - the total number of correct tokens in the answer given by the student;
- noOfTotalItems - the total number of tokens from which the answer is formed;
- grades - the grade obtained by the student after comparing the answer given by him with the correct answer;
- dates - the date on which the login session took place on the dsel platform.

The symbols corresponding to the selected AGLO are organized in a table built by a generator contained by the platform. The answer is divided into one or more tokens separated by space, depending on the exercise. Regarding the grade, it is calculated in direct proportion to the correct tokens the student wrote.

7.2. The domain-specific JavaScript Libraries

7.2.1. The JavaScript Library for Fractions Applications

For the exercises aimed at working with fractions to meet their goals, we have built a JavaScript library called Fractii.js. These contain the methods necessary for the notions concerned.

Both simple fractions and decimal fractions are generated randomly. For a simple fraction, the denominator and the counter are generated as random numbers. A simple fraction is initialized by the constructor function `randomFraction2()`. This constructor generates the fraction object based on the two random numbers the denominator and the counter.

For a mixed decimal fraction, are randomly generated:

- the integer part,
- the simple decimal part,
- the decimal periodic part.

A decimal fraction is initialized by the constructor function `randomFraction3()`. This constructor generates the fraction object based on a different number of random integers, depending on the fraction type. If you want to generate a simple decimal fraction, the periodic part will be set to zero. If you want to generate a simple periodic fraction, the non-periodic decimal part will be set to zero. Thus, we have the same constructor for a simple decimal fraction, a simple periodic fraction, or a mixed periodic fraction. These fractions have specific methods by which they are then transformed into ordinary fractions to be used in arithmetic operations.

```

function fractieAleatoare2 (nr, num)
{
    var f=newFractie ({ "numerator":nr, "numitor":num});
    return f;
};

function fractieAleatoare3 (pi, pzn, pzp)
{
    var f=newFractie ({ "parteIntreaga":pi, "parteZecimalaNeperiodica":pzn,
    "parteZecimalaPeriodica":pzp});
    return f;
};

```

Fig.7.4. The constructor functions for fractions objects

Thus both constructor functions return a "Fractie" fraction object who can call on their specific methods. The representation of constructor functions is captured in Figure 7.4.

The names of the constructor functions have a digit in their name. This digit represents the default number of parameters. These details help the tutor in case of reuse and debugging.

The type Fractie represents a fraction object.

Functions are also implemented for mathematical operations used in exercises such as addition, multiplication, division. Each of these functions has a parameter of the type fraction object. In this case, the mathematical operations are applied between the object that calls the function and the one that receives it as a parameter. All these functions return an object of the fraction type.

The function for calculating the power also has a parameter but in this case, this is an integer representing the exponent. And in this case, what is returned is a function type object.

Another function we have implemented is to generate the counter and denominator, this is useful for decimal numbers.

We also implemented functions that return a sign in the case of comparison and functions that return a word in the case of classification.

The most widely used function is the divisor function, which calculates how much the fraction must be simplified to become irreducible. This is called in 80% of the AGLOs.

7.2.2. The JavaScript Library for Intervals Applications

Regarding the AGLOs regarding the work with intervals, we have implemented the Intervale.js library.

The RandomInterval(*l*, *r*) constructor function will generate a closed interval object, where *l* and *r* are two randomly generated integers that represent the bounds of the interval. So, *l* is the left bound and *r* is the right bound.

The methods we have implemented in this library are:

- toString() - which transforms the interval object into a string so that it can be represented or compared;

- `intersection(a)` - realizes the intersection between the object `this` and the object `a` that it receives as a parameter;
- `union(a)` - realizes the reunion between the object `this` and the object `a` that it receives as a parameter;
- `difference(a)` - realizes the difference between the object `this` and the object `a` that it receives as a parameter.

These methods have been reused in AGLOs aimed at solving inequalities. We managed to achieve this reuse because the solution of an in-equation is an interval.

7.2.3. The JavaScript Library for Data Structures and Algorithms

To be able to create all the AGLOs FOR Data Structure and Algorithms discipline, we have created some JavaScript libraries. They individually target one of the desired topics, i.e. a sort, or an order. In the case of linked lists, however, it took several libraries to be able to obtain both a string shape and a graphic shape.

The JavaScript Libraries for Searching Algorithms

For the four search algorithms we made the following libraries:

- `BinarySearch.js`;
- `LinearSearch.js`;
- `InterpolationSearch.js`;
- `SentinelLinearSearch.js`.

The name of each is chosen suggestively for better reuse and ease in further development.

Each of these libraries contains a constructor function and a `getSteps()` function. The `getSteps()` returns the steps of the targeted algorithm, memorize as strings. Each stage is represented as one step at a time.

For the linear search, the library contains a constructor for initializing `linear_search` objects and a `getSteps()` function. The constructor function is called with two parameters, the first represents the string in which it is sought and the second the element that is sought.

The `getSteps()` function returns the steps of the algorithm, namely on each row is written the value of the index of the table that is compared with the searched element, the value of the element, and the result of the comparison that can be yes or no. If for example the index is 0, the value of `tab[0]` is 8, and the value of the searched element is 3, then the first step of the linear search will have the following representation: `i = 0 tab[0] == 3? 8 == 3? not`.

For the linear search with sentinel, the library contains a constructor for initializing `SentinelLinearSearch` objects and a `getSteps()` function. Because this algorithm is an improved version of the linear search algorithm, the functions used are similar.

The `BinarySearch.js` contains three functions:

- the constructor function with which the `binary_search` objects are initialized;
- the `getSteps()` function that returns a string containing the steps of the algorithm;

- the `belongs(element, array)` function that checks if an element belongs to an array and returns a Boolean value.

A step in the binary search algorithm has the following form: $s = 0, d = 5, m = 2, 20 > 16$, where s represents the position of the left element, d the position of the right element, m the position of the middle element, and 20 and 16 represent the values of `tab[m]`, respectively of the searched element.

For interpolation search, we created a library that contains a constructor function needed to initialize `interpolation_search` objects and a `getSteps()` function. Each step of the interpolation search is of the following form: $s = 0, d = 9, m = 3, 67 = 67$, where s represents the position of the left element, d the position of the right element, m the position of the pivot, and 67 and 67 represent the values of `tab[m]`, respectively of the searched element.

The JavaScript Libraries for Sorting Algorithms

Regarding the sorting algorithms, also in their case, we made libraries named suggestively with the name of the targeted algorithm.

For the sort algorithms, the `getSteps()` function aims to obtain all the steps of the algorithm, i.e. the intermediate variants through which the string passes during the sorting through the respective algorithm.

The `InsertSort.js` library and the `SelectSort.js` library contains two functions: a constructor function for initializing `insert_sort` objects, respectively `select_sort` objects and a `getSteps()` function that returns all intermediate variants of the string until it is sorted.

The `BubbleSort.js` contains the following functions/methods:

- the constructor function required to initialize `bubble_sort` objects,
- `getSteps()` function for obtaining algorithm steps,
- the `getFirst()` function which returns the first pair of elements to be exchanged within the algorithm,
- the `getNoOfChanges()` function that returns how many changes were made in a single traversal of a table in one step of the bubble sort algorithm.

The `ShellSort.js` contains the following functions:

- the constructor function required to initialize `shell_sort` objects,
- `getSteps()` function for obtaining the algorithm steps,
- the `getSeries()` function that returns all series of a table for a given gap h ,
- the `getFirstSerie()` function that returns the sequence of states for the first series that will be sorted independently for a given table and gap h .

For the quick sort algorithm, the specific library contains two functions: a constructor function for initializing `quick_sort` objects and a `getSteps()` function that returns all intermediate variants of the string until it is sorted.

The JavaScript Libraries for Linked Lists

Regarding the linked lists, we made five libraries for working with them.

There are three JavaScript libraries that we have made them represent a linked list and are used for both random and ordered linked lists:

- the `LinkedListLink.js` is useful for making graphical connections between list nodes,
- the `LinkedListNode.js` which contains a constructor function for initializing a `LinkedListNode` object, a function for graphically representing a node, and a function for representing the node as a string,
- the `LinkedListNodeShape.js` which contains everything needed to graphically represent the node of a list.

The other two libraries are made specifically for random linked lists and ordered linked lists.

The `RandomLinkedList.js` contains:

- a constructor function for initializing `random_linked_list` objects,
- three functions `fillWithIntegers()`, `fillWithFloats()` and `fillWithChars()` that randomly generate a list whose information field contains elements of integer, float or character type, respectively,
- `atoString()` function that returns the representation of the random linked list as a string,
- `acomputeSVG()` function that calculates the data needed to represent the existing random linked list,
- `atoSVG()` function that generates the graphical representation of the random linked list,
- `anadd(location)` function that returns a new list obtained by adding to the existing one a node in the position given by the `location` parameter,
- an `add_pos(location, poz)` function that returns a random list obtained by adding to the existing one a node in the `location` position, which can be before or after, referring to the element on the `poz` position in the list,
- `adelete(poz)` function that removes from the random linked list the item from the `poz` position,
- a `delete_pos(poz, option)` function which, depending on the value of the options parameter, deletes the element before or after the element on the `poz` position in the random linked list,
- `anode(poz)` function that returns the node that has the `poz` position in the random linked list.

The `OrderedLinkedList.js` contains:

- a constructor function for initializing `ordered_linked_list` objects,
- two functions `fillWithIntegers()` and `fillWithFloats()` that randomly generate an ordered linked list whose information field contains elements of integer or float type, respectively,
- `atoString()` function that returns the representation of the ordered linked list as a string,
- `atoSVG()` function that generates the graphical representation of the ordered linked list,
- `awhere_to_add()` function that returns the position where a node must be added for the linked list to remain ordered, namely at the beginning, at the end or inside,
- `anadd()` function that returns a new ordered linked list obtained by adding a node to the existing one.

For the random double linked lists, we made four libraries. Three of these, as in the previous case, are realized to have the necessary tools for the representations

of the lists, namely `DoubleLinkedListLink.js`, `DoubleLinkedListNode.js`, and `DoubleLinkedListNodeShape.js`.

The fourth `RandomDoubleLinkedList.js` library contains the following functions/methods:

- a constructor for initializing `random_double_linked_list` objects,
- `fillWithIntegers()` for randomly generating a random double linked list whose key fields contain integer information,
- `fillWithFloats()` for randomly generating a random double linked list whose key fields contain float information,
- `fillWithChars()` for randomly generating a random double linked list whose key fields contain character information,
- `toString()` to represent the double linked list as a string,
- `computeSVG()` for the calculation of the data necessary for the graphical representation of the double linked list,
- `toSVG()` for graphical representation of double linked lists,
- `add()` to add to the beginning or end of the double linked list depending on the desired option,
- `node(pos)` to return the node at the `pos` position in the double linked list,
- `add_pos(pos)` to add to the double linked list a node before or after the node located on the `pos` position,
- `delete(pos)` to delete the item from the `pos` position in the double linked list.

7.2.4. The JavaScript Library for Design and Analysis of Algorithms

To make these AGLOs that aim to work with trees, we made four libraries, each one being named in such a way as to highlight what it was created for.

The following libraries: `Circle.js`, `Line.js`, and `TreeNode.js`, have a role in making the graphical representation of the tree. In the `Circle.js` the functions necessary for the representation in the form of a circle of a node of a structure are found. In the `Line.js` library, you will find the necessary functions for making arches between nodes. These libraries are generally made to be able to be used to represent other structures than trees, namely graphs.

The fourth library is called `Tree.js` and contains all functions specific to working with trees in the scenarios we imagined. Functions are:

- the constructor function which generates a tree according to the parameters set in the object argument,
- the function `toSVG()` which generates the graphical representation of the tree,
- the function `height()` which calculates and returns the height of the tree,
- the function `index(key)` which returns the order of a node knowing its key,
- the function `getKey(index)` which computes the key of a given index,
- the function `getNodeDegree(index)` which computes the node degree of a given index,
- the function `allNodesDegrees()` which returns as a string all the nodes of the tree and their degrees,

- the function *getArrayKeys()* which returns the tree keys as an array,
- the function *getChildParent()* which returns as a string all the pairs of shape: node - father node in the tree,
- the function *tree_parentIdList()* which computes the list of parent ids,
- the function *tree_firstChildIdList()* which computes the list of first child applied for each node,
- the function *tree_levels()* which computes the multiline representation of a tree by levels,
- the function *inorder()* which returns the traversal of the tree in inorder,
- the function *preorder()* which returns the traversal of the tree in preorder,
- the function *postorder()* which returns the traversal of the tree in postorder.

7.2.5. The JavaScript Library for Operating System

In order to be able to create all the AGLOs presented in this chapter, we have created several JavaScript libraries.

The name of each is chosen suggestively for better reuse and ease in further development.

The File.js library contains the following functions: the constructor function, the setRandomName function, and the toString function. The constructor function has the role of creating a tab object. The setRandomName function initializes a file name out of 26 possible, followed by a natural two-digit number with an extension of ten possible variants. Such a combination guarantees a great diversity of possible names, over twenty-five thousand. The toString function transposes the file name in string format so that it can be represented in applications.

The Folder.js library is made on the same principle as the previous one. It contains three functions:

- the constructor function has the role of creating a folder object,
- the setRandomName function initializes a folder name out of 26 possible, followed by a natural two-digit number, thus providing over two thousand five hundred possible names,
- the toString function transposes in string format the folder name so that it can be represented in applications.

The FolderTree.js library contains the following functions:

- the constructor function that performs a hierarchy of folders up to five levels,
- the preorderOnNodes() function, to generate the display of the preorder folder tree using * and nodes,
- toMkdirPathsString() function, iterates folder paths and chains the mkdir command,
- toMkdirCdPathsString() function, to create individual folders,
- toTouchPathFilesString(), to populate the folder tree with files using the touch path on the folder paths,
- the toPathFilesString() function, to populate the folder tree with files using the touch command that moves in each folder,

- the `iterateTerminalNodes()` function, to iterate the pre-command terminal node folders and store each path in the tree returns in the path argument the path-filled array,
- the `iterateAllNodes()` function, to iterate all nodes in the pre-order,
- the `toString()` functions for each of the above to translate the result into a form that can be returned to the student, namely as a string.

7.3. Summary

In this chapter, we have presented the DSEL platform where students can access the AGLOs made by us. They can log in to the platform with a G-mail address or a Facebook account.

Next, we presented the domain specific JavaScript libraries that we have created in order to be able to create the AGLOs presented in chapters 4 and 5.

8. VALIDATION OF AUTO-GENERATIVE LEARNING OBJECTS

8.1. Case Study for Fractions AGLOs

T-tests are basic tests for the analysis of continuous data. We chose to use a T-test to determine if there is a significant difference between a student's classical grading and the grading with the help of our AGLOs.

A t-test allows us to compare the average values of the two sets of notes and determine if they have major differences between them.

The test takes a sample from each of the two sets and establishes the statement of the problem by assuming a null hypothesis that the two means of evaluation are similar. Based on the applicable formulas, certain values are calculated and compared with the standard values, and the assumed null hypothesis is accepted or rejected accordingly.

Calculating a t-test requires three key data values. These include the difference between the mean values from each data set, the standard deviation, and the number of data values.

To perform the statistical analysis regarding the application of the AGLOs targeting middle school arithmetic, we applied them to a sample of 12 students.

Student ID	Grades using AGLOs	Grades in classical teaching
1	9	10
2	7.75	8
3	5.3	7
4	7	9
5	5.5	6
6	8.3	8
7	8	8
8	7.5	7
9	8	8
10	4	5
11	10	9
12	5.5	6

Table 8.1. Grades obtained in the two types of assessments

Table 8.1 shows the grades obtained by students in the classical version, as well as after the use of these auto-generative learning objects.

The F score is a measure of the accuracy of a test. The first thing to do is to state the affirmation of the null hypothesis, namely there is no significant difference between the two evaluations.

The F score is often used in machine learning [5]. F-measure is the harmonic mean of recall and precision. The precision is the fraction of relevant instances among

the retrieved instances, while recall is the fraction of the total amount of relevant instances that were retrieved.

	True Positive	True Negative
	11	0
	0	1
	False Positive	False Negative

Fig.8.1. Contingency table: false positive, false negative, true positive, and true negative values of the grades

In Figure 7.1 we divided the marks obtained in an error matrix. We will discuss the possibility that the use of AGLOs does not distort the reality of learning assessment.

A p-value of less than 0.05 will indicate that there is less than a 5% chance that a significant change in grades will be due to the use of learning objects in teaching. Therefore, there is a 95% chance that learning objects will effectively support the learning process.

According to the collected data, the average decrease obtained is 0.42.

From the standard deviation, one can calculate the standard error (SE) this being 0.25.

The value of t is the test statistic of the t-test and is calculated as follows $t = 0.42 / 0.25 = 1.68$.

The p-value is less than 0.002.

We can conclude that the application of generative auto-learning objects leads to a real evaluation of the student.

In terms of positive identification, it can be calculated the precision which in our case is equal to 1.

To identify what proportion of real positives has been correctly identified we will calculate the recall which in our case is equal to 0.91.

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. In our case, according to the data presented in Figure 8.1, the accuracy comes out to 0.91, or 91%.

The purpose of the questionnaire applied to sixth graders who used learning objects made by us was to find out the students' willingness to use the online environment in their learning activity and to determine the degree of satisfaction offered by the interactive objects.

The questionnaire was composed of 10 questions structured on two existing areas at the individual level of the possibility to learn online from home, the impact of auto-generative learning objects.

Next, we will analyze the results obtained for each area of interest.

Regarding the possibility of each student using a gadget in the learning process, most students said that 75% use the tablet, laptop, or phone in learning activities, while 25% do not use them.

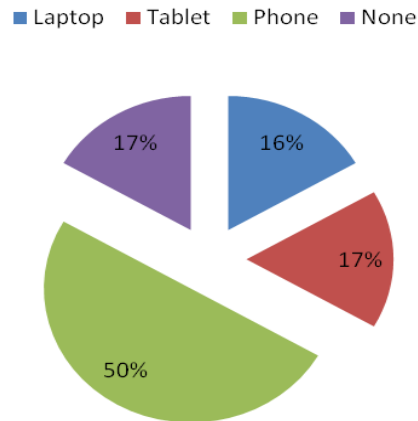


Fig.8.2. The degree of use of gadgets by students

Regarding the degree of appreciation on self-generative learning objects, the result of the questionnaire was the following: a significant percentage of 67% of the respondents said that this approach is interesting and that they would like teachers to integrate them into their e-learning materials, and 8% of them considered it not useful in their learning process.

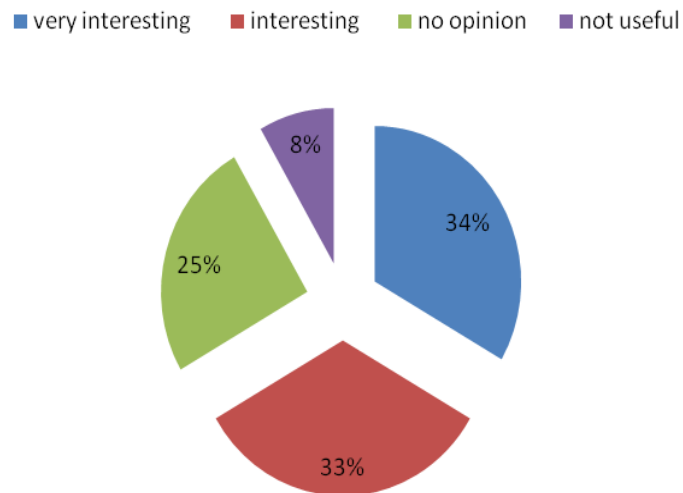


Fig.8.3. The degree of interest of the students on the LOs

Following the diagrams above, it can be seen that currently the learning activity is carried out in the online environment, it is based on independent activity and the dynamic objects are appreciated by the students.

8.2. Case Study for Intervals, Equations and Inequations AGLOs

In this section, we will present the statistical analysis of the results obtained for a group of 8th-grade pupils. We applied a test with these 9 AGLOs on a group of 50 pupils. Also, to the same group of pupil, we applied a classic test.

Analyzing the data obtained, we found that the difference between marks is over 1 in a single case, and we obtained identical grades in 24 of the cases. We apply the t-test for assessing the statistical significance of the difference between the two evaluations.

The hypothesis proposed for evaluation is: the chances that these learning objects do not really reflect the level of knowledge acquired by pupils are less than 5%.

After comparing the marks obtained from a classical test with those obtained by evaluation with AGLO, we had as a result the standard deviation (SD) having a value of 0.75, and the standard error (SE) having a value of 0.11. According to the values obtained, the value of t , which measures the size of the difference in relation to the variation of our data, is 1,88. The t distribution probability percentage, the p -value is 0.03. A p -value less than 0.05 means that there is no significant difference between the marks obtained by the two assessments.

In the evaluation of the pupils, we applied a test consisting of nine AGLOs having as learning objectives:

- i. Calculate intervals intersection;
- ii. Calculate intervals reunion;
- iii. Calculate intervals difference;
- iv. Solving modulus based equation;
- v. Solving modulus based inequation;
- vi. Solving inequation of different forms.

A group of 50 pupils answered the AGLOs and fulfilled a face-to-face test on the previously listed learning objectives.

In order to determine the accuracy of the approach, we will compare the two result sets. We focus on the pupil pass/fail classification accuracy. The grades under 5 mean failure and grades above 5 mean passing. We study the confusion matrix (see Figure 8.4) obtained by comparing the grades from AGLOs based assessment with the grades from the face-to-face assessment.

In creating the confusion matrix, we considered the following:

- false negative (FN) - a pupil who in the classical evaluation has a grade below 5 and in the evaluation with AGLOs has a grade above 5;
- false positive (FP) - a pupil who in the classical evaluation has a grade above 5 and in the evaluation with AGLOs has a grade below 5;
- true negative (TN) - a pupil who in the classical evaluation has a grade below 5 and in the evaluation with AGLOs he also has a grade below 5;
- true positive (TP) - a pupil who in the classical evaluation has a grade over 5 and in the evaluation with AGLOs he also has a grade over 5.

Evaluation with AGLOs

		Under 5	Over 5
Classical evaluation	Under 5	3	0
	Over 5	1	46

Fig.8.4. The confusion matrix

Having established these, we can calculate the accuracy. Accuracy is calculated according to the formula [68]:

$$\frac{TN + TP}{TN + FN + TP + FP}$$

Therefore, the accuracy of the approach has a value of 98% for the pass/fail hypothesis. This value is a high enough number for the model to be considered 'accurate'.

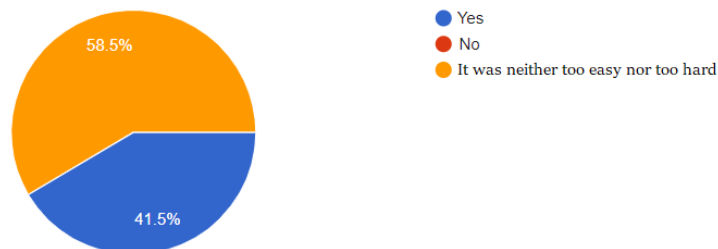
However, we will calculate further the precision and the recall. The precision or positive predictive value is defined as the percentage of TP number out of the number of all positives (TP + FP). In our case, the precision has the value 1. This value is very good because it indicates that any pupil who has a grade above 5 in this assessment, means that he is indeed a pupil whose level of knowledge is above grade 5.

The recall or the true positive rate is defined as the percentage of TP number out of TP + FN. The recall has a value of 0,978. This measure tells us how well we can identify pupils who have not accumulated the necessary knowledge to obtain a grade above 5. So, with the help of this model, we can identify pupils with poor knowledge with a correctness of about 98%.

F1 score is the harmonic mean of precision and recall. In our study, the F1 score has a value of 0,989. We consider this to be a good result.

2. It was easy to use these interactive questions (AGLOs)?

41 responses



3. You liked these interactive questions (AGLOs)?

41 responses

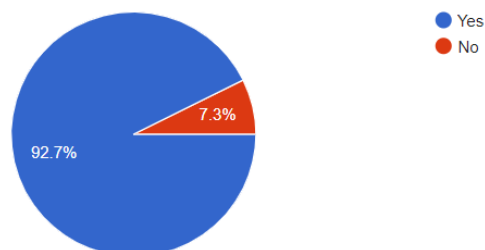


Fig.8.5. The questionnaire answers to the second and third questions

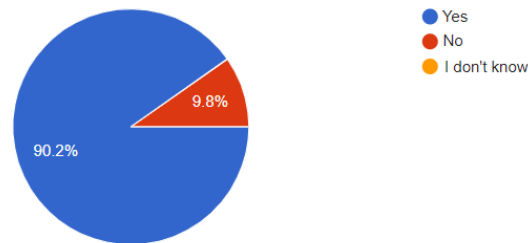
In order to receive feedback from pupils on the appreciation of these types of exercises, we chose to apply a satisfaction questionnaire. 41 pupils out of the 50 who used AGLOs answered the questionnaire. The questionnaire includes six closed questions and one open question. A set of questions was designed to investigate the usage of computing devices by the pupils. It turned out that 75% of them are using laptops or desktop computers.

In the followings, we will analyze only the questions that refer strictly to AGLOs and their use. Pupils were asked if it is easy for them to access and use these types of exercises. In this question, 17 of the interviewed pupils considered that the AGLOs are very easy to use, and 24 of the pupils considered that they are neither difficult nor easy to use (see Figure 8.5). Regarding their appreciation, 7% of the pupils did not like the AGLOs, while 93% were very excited about them.

Regarding their usefulness, pupils were asked if they found them useful for the process of knowledge accumulation or in the assessment. To this question, 61% considered that they would be more useful in the evaluation, and 34% considered that they would be more useful in practicing the Arithmetic notions.

6. You want to have more learning activities with such interactive questions (AGLOs)?

41 responses



7. You would like to use interactive questions (AGLOs) at other disciplines?

41 responses

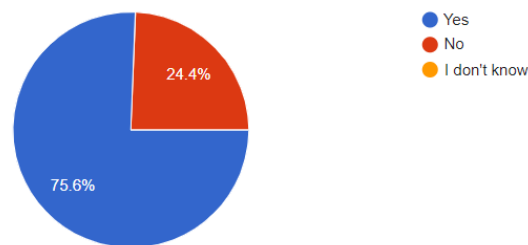


Fig.8.6. The questionnaire answers to the last two questions

In the open-ended question, pupils were asked: what do they think should change in these exercises. Four pupils felt that the way their answer was assessed could be changed, as they encountered problems in not following the punctuation.

Regarding the pupils' desire to use AGLOs, after this experiment, we found out that 90% of the pupils would like to interact with them, but only 75% would like them for other disciplines (see Figure 8.6).

Since no pupils considered difficult to use these types of exercises, we can conclude that they represent accessible learning materials and that they delighted by them. In addition, they consider AGLOs useful and express their desire to have more learning activities in which to use them.

8.3. Summary

In this chapter, we have presented two case studies of using AGLOs.

First, we used the AGLOs to deepen and evaluate the notions related to fractions in the sixth grade. We evaluated the model comparing the evaluation with the marks obtained by the students at a classical evaluation. After calculating the F score of the test, we can say that the evaluation with AGLO reflects the real value of the level of acquisition of the student's knowledge.

Next, we applied in the teaching-evaluation process to the eighth grade AGLOs that refer to operations with intervals and inequalities of different types. To determine the accuracy of the model and in this case we compared the grades obtained by students in two tests, one face-to-face and one using AGLOs. In this case, the result was good.

We also applied satisfaction questionnaires to students to get feedback. Following the answers received, we can conclude that AGLOs are a type of exercises appreciated by students, and which they desire in the instructive-educational process.

9. CONCLUSIONS AND PERSPECTIVES

9.1. Conclusions

In the state of the art of this thesis, we can conclude that learning and teaching in education have faced significant challenges. Designing learning experiences that successfully integrate digital tools is still a real issue. The use of AGLOs in the teaching-evaluation process could be one of the solutions.

In order to design LOs, we carried out a review of the LO literature, an inventory, and a comparison of the existing approaches. LOs have several definitions and models designed in different directions: pedagogical, technical, and economic.

The IEEE model was made to facilitate the use and distribution of LOs. SCORM facilitates the provision of a very general model on which a certain type of LO can be built and offers a high level of flexibility because it does not require the organization of the content in a certain way. Cisco was developed as a strategy for developing and implementing reusable learning objects (RLOs). The Dublin Core model provides a simpler set of elements that overlap with IEEE LOM and is useful in sharing metadata for the specific needs of the educational community. The Learnativity content model was developed as a model with a fixed number of granularity levels and is a basis for extending the learning content architecture. NLOs were created to operate on the National Education Training Group's own LMS prototype. ALOCoM was built as a model that allows the generalization of other standards such as Learnativity, SCORM, or NETg. H5P is an open-source tool used to facilitate the creation of various interactive activities that can be integrated into different sites or platforms.

The sequencing of LOs is a method designed to give students individualized material. LOs are seen as small reusable learning units that can be grouped to get a lesson or a course by using artificial intelligence techniques. A different theory proposed for LOs to be more efficient is to improve them with the help of notions of object-oriented design, so the OOGLOM model was developed. Another approach is the one used for the implementation of the CLAVIRE e-Science platform as a basis for hybrid learning resources (HLR), these hybrid objects being mapped and represented as a cloud package.

From the studied materials, we realized that we can improve teaching and learning through the use of ICT. However, to achieve maximum impact, a new generation of LOs was needed. The basic idea of the GLO concept is to achieve a successful LO with the ability to reuse, not necessarily focusing on the content. The GLOs designed by Tom Boyle are design models implemented as rich reusable pedagogical templates. Another approach to GLOs was that of Damasevicius and Stukys, namely a knowledge-based model expanded with meta-programming technologies.

In the study, we did for the thesis we also focused on two of the Moodle plugins, namely Moodle Coordinate Questions and Moodle Calculated Questions. They use the GLO principles by creating a template that contains static text blended with variables. The variables are replaced at instantiation time with randomly generated values in a certain interval.

Based on the comparison of the existing LO models, we concluded that the advantages that AGLOs offer motivates us researching their design and implementation. Based on the comparisons we can conclude that AGLOs are templates accessible from any device, which provides feedback and automatic assessment. They are dynamic exercises because they are made of blending static text with variables, these variables being automatically instantiated with random values at each instantiation. They can be used both in the evaluation and in the learning process for practicing STEM concepts.

In order to create an AGLO, it is necessary to follow a few steps that we have grouped into an algorithm. The tutor must choose a learning objective, to carry out an exercise that sees that objective. In the created exercise, the tutor has to identify the input data, the intervals in which they could be randomly generated, to identify the intermediate data, their type, and if a specific function is needed for the calculation. The tutor must identify the algorithmic steps in which the answer is calculated. He must also choose the intervals according to the desired degree of difficulty for the exercise. He must perform object instantiation testing to determine any non-compliant or incorrect cases that may occur.

The AGLO model is defined using the EBNF meta-language. From a structural point of view, each ALGO consists of six sections: name, scenario, theory, question, answers, and feedbacks.

Regarding the disciplines we approached, we can conclude that AGLOs are pedagogical models that can model several concepts in the area of STEM disciplines. We managed to apply this model to disciplines like: Middle School Arithmetic, Data Structures and Algorithms discipline, Algorithm Analysis, and Design discipline, and System Operating discipline.

For the Arithmetic discipline, we chose to approach the subject of fractions where we designed AGLOs for an entire chapter from the manual of the fifth-grade curriculum. Regarding the fractions, we made AGLOs dealing with the classification of fractions, amplification and simplification of fractions, operations with fractions, transformations of fractions, and addition of integers in the fraction, removal of wholes from the fraction. We also addressed eighth-grade Arithmetic on interval operations, solving modulus equations, solving modulus inequations, solving several types of inequalities, and applying abbreviated calculation formulas and factor decompositions according to formulas.

For the Data Structures and Algorithms discipline, we have made AGLOs for several types of searches and different sorting methods, namely insertion search, selection search, bubble sort, shell sort, and quicksort. We have also developed AGLOs aimed at working with simple linked linear lists and double linked linear lists. In the case of lists, the exercises aimed at recognizing the successor or predecessor of an element, adding a node before or after a given element, adding it to an ordered list, deleting a node according to a certain criterion.

For the Algorithm Analysis and Design discipline, we created AGLOs regarding the notions of tree and graph. These are aimed at the correct knowledge and proper use of specific notions, at traversing trees in preorder, postorder, and inorder, at constructing the adjacency matrix and the adjacency structure corresponding to a graph.

For the System Operating discipline, we have created AGLOs aimed at knowing the commands for working with folders, the commands for working with files, the commands for controlling processes, the commands for partitioning the disk, the administrative commands, as well as some basic network commands.

Thus, we managed to create 34 AGLOs targeting arithmetic notions, 97 AGLOs for Data Structures and Algorithms discipline, 17 AGLOs for Algorithm Analysis and Design discipline, and 27 AGLOs for Operating Systems discipline.

To fulfill the imagined learning scenarios we implemented domain specific JavaScript libraries. Thus, we developed five libraries modeling 42 reusable concepts containing the necessary functions and methods.

In order for students to be able to access the AGLOs, we use a web platform. Students can access the platform using a Gmail or Facebook account.

Regarding the AGLO application in practice, we conducted two case studies for the fifth and eighth grades.

In the first case study, we applied in the learning-assessment process of 12 pupils, AGLOs aimed at working with fractions. The hypothesis proposed by us is that the grading of students using AGLO will reflect the value of the acquisition reached by the student. According to the collected data, the calculated p-value was less than 0.002. Thus, our hypothesis has been achieved.

In the second case study, we applied a set of nine AGLOs for the evaluation process of 50 eighth-grade pupils. The hypothesis proposed by us is that the chances that these learning objects do not reflect the level of knowledge acquired by pupils are less than 5%. In this case, based on the collected data, the calculated p-value was 0.03.

From the surveys, we learned that the students were very interested in these types of learning objects and showed their openness to do activities that use AGLOs.

We can summarize that AGLOs are templates easy to use by students, accessible from any device, which provides feedback and automatic evaluation. They can be used both in the evaluation process and in the learning process for practicing STEM concepts.

We consider that AGLOs are learning tools that could help us to increase the quality of online education, by providing feedback to the student and automatic evaluation of the response.

9.2. Meeting the Objectives

The doctoral research performed in this theses had as main goal the development and implementation of an AGLO-based approach meant to facilitate learning and automatic assessment for STEM disciplines. This doctoral thesis is in the sphere of general interest concerns in the field of eLearning, improving the quality of education through the integration of ICT. The treated topic represents a complex and topical scientific approach with a strong interdisciplinary character, being able to be used in the educational activity.

In an exhaustive enumeration, the personal contributions brought by the author, following the research carried out during the doctoral studies, are the following:

1. We developed an abstraction algorithm to create AGLO templates. In this sense, eleven steps have been merged that can be followed by any teacher to create reusable AGLO templates. The abstraction-based methodology was published in [69].

2. We made 175 AGLOs on several notions from STEM disciplines to show the wide applicability of this model. In this sense, we have implemented the following:

- 21 AGLOs for working with fractions, in fifth grade, published in [70];
- 13 AGLOs for eighth-grade Arithmetic, published in [71];
- 97 AGLOs for Data Structures and Algorithms discipline, published in [72];
- 17 AGLOs for Algorithm Analysis and Design discipline, published in [73];
- 27 AGLOs for Operating Systems discipline, published in [74];

3. A number of five domain-specific JavaScript libraries was developed to model the targeted concepts. These libraries contain the functions and methods necessary for the implementation of the proposed AGLOs.

4. We conducted a study on a group of 12 fifth graders to investigate the effectiveness of learning compared to classical approaches. In this sense, we used AGLOs in the process of learning and evaluating the notions related to fractions. We used specific metrics to compare the results obtained in the evaluation with AGLOs and in the classical evaluation. Statistical data obtained were published in [70].

5. We conducted a study of a group of 50 8th graders to determine whether the AGLO's automatic assessment mechanisms are very close to the tutors' assessment. In this sense, we applied to the class two evaluations, a classical evaluation and an evaluation using AGLO. Statistical data obtained using specific metrics were published in [75].

The researches undertaken during the doctoral studies allowed the elaboration and publication, as first author, of a number of 8 articles published in the volumes of some international scientific conferences, indexed by ISI, which are found in the bibliography.

9.3. Future Work

As future work, we would like to extend our approach to a larger number of STEM disciplines and their concepts. This implies abstracting generic models from exercises imagined for the fundamental concepts from:

- Arithmetic, notions from 6th and 7th-grade curriculum;
- Geometry, notions like triangle, square, rectangle, circle - specific elements, calculations of areas and perimeters;
- Medicine schools disciplines like statistics notions and some formulas applied in the high school biology;
- Mechanics, formulas for university courses and application at seminars and laboratories;
- Civil constructions, formulas, and notions for university courses and application at seminars and laboratories;
- Chemistry, reaction equations, formulas, and notions for high school curriculum;
- Physics, formulas, and notions for high school curriculum.

Another direction is to develop a way to integrate AGLO into an LMS or to make a mobile application for these objects.

Another challenge is to make these designs accessible to tutors through a tool that permits the creation and adaptation of AGLOs. An AGLO designer could have a friendly graphical interface that allows tutors to create and/or modify their objects. It could also provide a translation into several languages such as English and French.

The correct use and creation of AGLOs would also require the development of tutorials. Tutorials could be made on several levels depending on the needs of the tutors: just use, modify existing objects, and create your own objects.

Another direction that publishes the code/prototype with an open-source license.

We consider that our model is at the TRL4 [58] level because it was tested in a laboratory environment and two practical situations. Through the future activities that we propose in our future work, we consider that it will move to TRL5, as we will be able to complete other relevant validations in the practical environment.

BIBLIOGRAPHY

- [1] M. White, "Synthesis of research on electronic learning," *Educational Leadership*, no. 40(8), p. 13-15, 1983.
- [2] T. S. C. Learning, Draft Standard for Learning Object Metadata. IEEE Standard 1484.12.1, New York: Institute of Electrical and Electronics Engineers, 2002.
- [3] "Advanced Distributed Learning Initiative, SCORM, [Online]," Available: <https://www.adlnet.gov/adl-research/scorm> [Accessed: 28-05-2018], 2001.
- [4] LTSC, "Ieee standard for learning object metadata. Technical report, IEEE," [Accessed: 28-05-2018], 2002.
- [5] "IMS Learning Design," Available: <http://www.imsglobal.org/learningdesign/index.html> [Accessed: 20-01-2021], 2003.
- [6] "Learning Resource Metadata Initiative," in *Available: https://www.dublincore.org/specifications/lrmi/1.1/* [Accessed 18-04-2018].
- [7] C. Systems, Reusable Information Object Strategy. Version 3.0, 1999.
- [8] B. Davis, C. Colleen and E. Wagner, *The Evolution of the LMS: From Management to Learning*, Sage Road Solutions, LLC, 2009.
- [9] Moodle, Releases - MoodleDocs, Available: docs.moodle.org. [Accessed 18-04-2018].
- [10] M. Kunkel, *The official ILIAS 4 practical book*, München : Auflage. Addison-Wesley, 2011.
- [11] „ About Canvas | Edtech Learning Platform | Instructure,” 2013.
- [12] C. B. Chirila, "Auto-generative learning objects in online assessment of data structures disciplines," in *BRAIN - Broad Research in Artificial Intelligence and Neuroscience*, Bacau, Romania, 2017.
- [13] D. Robertas and Š. Vytautas, "On The Technological Aspects of Generative Learning Object Development," *Springer*, 2008.
- [14] T. Boyle, „Design principles for authoring dynamic, reusable learning,” 2003.
- [15] T. Boyle, „The desing and developement of second generation learning objects,” World Conference on Educational Multimedia, Hypermedia & Telecommunications, Orlando, 2006.
- [16] T. Boyle, "Generative learning objects (GLOs): design as the basis for reuse and repurposing," London Metropolitan University, London, 2009.
- [17] R. Damasevicius and V. Stukys, "Specification and Generation of Learning Object Sequences for E-learning Using Sequence Feature Diagrams and Metaprogramming Techniques," 2009.
- [18] Š. Vytautas and D. Robertas, "Towards knowledge-based generative learning objects," *Information Technology and Control*, vol. 36, no. n2, 2007.
- [19] W. Hodgins, "The Future of Learning Objects," Vols. Vol. 46, No. 1,, 2006.
- [20] A. Chiappe, Y. Sergovia and H. Rincon, "Toward an instructional design model based on learning objects," *Education Tech Research Dev*55:671-681, 2007.

- [21] T. Boyle, J. Cook, R. Windle, H. Wharrard, D. Leeder and R. Alton, "An agile method for developing Learning Objects," Paper presentation, ASCILITE conference, Sydney, Australia, 2006.
- [22] D. Rehak and R. Mason, "Engaging with the Learning Object Economy," Littlejohn, Alison, Reusing Online Resources: A Sustainable Approach to E-Learning, pp. 22-30, London, 2003.
- [23] J. San Diego, D. Laurillard, T. Boyle, C. Bradley, D. Ljubojevic, T. Neumann and D. Pearce, "Towards a user-oriented analytical approach to learning design," *ALT-J, Research in Learning Technology*, vol. 16, no. 1, pp. 15-29, 2008.
- [24] "Advanced Distributed Learning Initiative, Experience API, [Online]," Available: <http://adlnet.gov/adl-research/performance-tracking-analysis/experience-api> [Accessed: 28-05-2018], 2011.
- [25] T. Boyle, "Layered learning design: Towards an integration of learning design and learning object perspectives," *Computers & Education*, vol. 54, nr. n3, 2010.
- [26] M. Gruene, K. Lenz and A. Oberweis, Pricing of Learning Objects in a Workflow-Based E-Learning Scenario, The 38th Hawaii International Conference on System Sciences, 2005.
- [27] "IMS specifications," 2014.
- [28] M. I. Dublin Core, "Innovation in metadata design, implementation & best practices," Available: <http://dublincore.org/documents/dces/> [accessed: 19-07-2018], 2010.
- [29] E. D. Wagner, Steps to Creating a Content Strategy for Your Organization, The eLearning developers' journal, October 2002.
- [30] C. Allen and E. Mugisa, "Improving Learning Object Reuse Through OOD: A Theory of Learning Objects," *Journal of Object Technology*, ETH Zurich, 2010.
- [31] J. L'Allier, Frame of Reference: NETg's Map to Its Products, Their Structures and Core Beliefs, Whitepaper, 1997.
- [32] K. Verbert and E. Duval, "ALOCOM: a generic content model for learning objects," *Int J Digit Libr* 9:p. 41-63, 2008.
- [33] A. C. A. Buccella and N. Brisaboa, An ontology approach to data integration, vol. 3, *Journal of Computer Science & Technology*, 2003.
- [34] C. Ullrich, The learning-resource-type is dead, long live the learning-resource-type!, *Learn. Objects Learn. Des.* 1(1), 7-15, 2005.
- [35] W. Ceusters and L. Bouquet, Language engineering and information mapping in pharmaceutical medicine, *J. Belg. Med. Inform. Assoc.* 7(1), 26-34, 2000.
- [36] Z. Homanová and T. Havlásková, H5P interactive didactic tools in education, Palma, Mallorca, Spain: EDULEARN19 Conference, 2019.
- [37] H5P, "H5P documentation".
- [38] R. Ellis, Field Guide to Learning Management, ASTD Learning Circuits, 2009.
- [39] Desire2Learn, "Blackboard Digital Content," <http://www.desire2learn.com/>, Online; accessed 22-Aug-2020.

- [40] R. Jones and T. Boyle, "Learning Object Patterns for Programming," Informing Science Institute, London, 2007.
- [41] T. Boyle și C. Bradley, „ User Guide for the GLO Maker 2 Authoring Tool,” 2009.
- [42] M. A. K. Halliday, Learning how to mean., UNESCO: In Lenneberg E.H. (Ed.) Foundations of language development, 1975.
- [43] T. Boyle and A. Ravenscroft, Context and deep learning design, Elsevier, Computers & Education 59, 2012.
- [44] V. Štuikys, R. Burbaite and R. Damasevicius, "Teaching of Computer Science Topics Using Meta-Programming-Based GLOs and LEGO Robots," *Vilnius University*, vol. 12, no. No. 1, 2013.
- [45] R. Damaševičius, "Towards Empirical Modelling of Knowledge Transfer in Teaching/Learning Process," DOI: 10.1007/978-3-319-11958-8_29, 2014.
- [46] V. Stuiikys, R. Burbaite, K. Bepalova, T. Blazauskas and D. Barisas, "Stage-Based Generative Learning Object Model for Automated Content Adaptation," *Baltic J. Modern Computing*, vol. 5, no. No.2, 2017.
- [47] B. C. Chirila, "A comparison of MCQ and AGLO generative learning object models," *Annals of the Faculty of Engineering Hunedoara*, vol. 13, no. 4, pp. 37 - 41, 2015.
- [48] L. de-Marcos, J.-J. Martínez and J.-A. Gutiérrez, "Swarm Intelligence in e-Learning: A Learning Object Sequencing Agent based on Competencies," GECCO'08, Atlanta, USA, 2008.
- [49] P. Brusilovsky, „Adaptive and Intelligent Technologies for Web-based Education,” Special Issue on Intelligent Systems and Teleteaching, 4.19-25, 1999.
- [50] A. Claudine and E. Mugisa, "Improving Learning Object Reuse Through OOD: A Theory of Learning Objects," *Journal of object technology*, 2010.
- [51] K. Bochenina, A. Dukhanov, M. Karpova and V. Shmelev, "An approach to hybrid learning resource design for training professionals in Computational Science," *Procedia Computer Science*, vol. 101, no. DOI: 10.1016/j.procs.2016.11.051, pp. 439 - 448, 2016.
- [52] K. Knyazkov, S. Kovalchuk, T. Tchurov, S. Maryin and A. Boukhanovsky, "CLAVIRE: e-Science infrastructure for data-driven computing," *Journal of Computational Science* 3, 2012.
- [53] Sergey V. Kovalchuk, Pavel A. Smirnov, Sergey S. Kosukhin și Alexander V. Boukhanovsky, „Virtual Simulation Objects Concept as a Framework for System-Level Simulation,” nr. DOI: 10.1109/eScience.2012.6404413, 2012.
- [54] A. Najat, "The importance of statistical tools for data evaluations.," DOI: 10.13140/RG.2.2.34553.19042, 2020.
- [55] Hypothesis Testing, https://courses.edx.org/c4x/UTAustinX/UT.7.01x/asset/Chapter_12.pdf 16.
- [56] Wasserstein, Ronald and Lazar, Nicole, "The ASA's Statement on p-Values: Context, Process, and Purpose," *The American Statistician*, vol. 70, pp. 129-133, 2016.

- [57] P. Pichot, *Les Tests mentaux*, Presses universitaires de France, 1991.
- [58] NASA, "Technology Readiness Level," https://www.nasa.gov/directorates/heo/scan/engineering/technology/technology_readiness_level, [Accessed: 15-05-2021].
- [59] I. ECMA, Standard ECMA-262 ECMAScript Language Specification 11th Edition, <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>, 2020.
- [60] C. B. Chirila, H. Ciocarlie and L. Stoicu-Tivadar, "Generative Learning Objects Instantiated with Random Numbers Based Expressions," in *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, 2010.
- [61] Cicu, I., David, E., Iacob, I. and Ceuca, R., *Mathematics - 8th grade*, Bucharest: Intuitext, 2020.
- [62] V. Cretu, *Structuri de date si algoritmi. Notite de curs*, Universitatea Politehnica Timisoara, Facultatea de Automatica si calculatoare, 2020.
- [63] T. Cormen, C. Leiserson și R. Rivest, *Introduction to Algorithms*, ISBN 0-262-03141-8: The MIT Press, 2000.
- [64] D. Knuth, *The Art Of Computer Programming 3rd ed*, Boston: Addison-Wesley Professional, 2011.
- [65] D. E. Knuth, *Art of Computer Programming, Volume 3: Sorting and Searching (2nd Edition)*, Addison-Wesley Professional, 2 edition, 1998.
- [66] D. Knuth, *Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd Edition)*, Addison-Wesley Professional, 1997.
- [67] V. Cretu, *Proiectarea si analiza algoritmilor. Notite de curs*, Universitatea Politehnica Timisoara, 2020.
- [68] A. Tharwat, „Classification assessment methods: a detailed tutorial,” în *Applied Computing and Informatics*, 2018.
- [69] F. M. Costea, C. B. Chirila and V. I. Cretu, "Redesigning educational tools using auto-generative learning objects," in *ELearning and Software for Education (ELSE)*, Bucharest, 2019.
- [70] F. M. Costea, B. C. Chirila and V. I. Cretu, "Auto-Generative Learning Objects for Middle School Arithmetic," in *International Scientific Conference eLearning and Software for Education*, Bucharest, Romania, 2018.
- [71] F. M. Costea, C. B. Chirila and V. I. Cretu, "Middle School Arithmetic Auto-Generative Learning Objects to Support Learning in the Covid-19 Pandemic," in *In Proceedings of the IEEE 13-th International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, 2021.
- [72] F. M. Costea, B. C. Chirila and V. I. Cretu, "Auto-Generative Learning Objects for Learning Linked Lists Concepts," in *International Symposium on Electronics and Telecommunications*, Timisoara, Romania, 2020.
- [73] F. M. Costea, B. C. Chirila, O. S. Chirila și V. I. Cretu, „On the Generation of Random Data for Auto-Generative Learning Objects,” în *IEEE 13th International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, Romania, 2019.

- [74] F. M. Costea, C. B. Chirila and V. I. Cretu, "Towards Auto-Generative Learning Objects for Industrial IT Services," in *The IEEE 12-th International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, Romania, 2018.
- [75] F. M. Costea, C. B. Chirila și V. I. Cretu, „A Use Case for Arithmetic Auto-Generative Learning Objects in Pandemic,” în *Else*, Bucuresti, 2021.
- [76] R. Burbaite, K. Bepalova, R. Damasevicius and V. Stuikeys, "Context-Aware Generative Learning Objects for Teaching Computer Science," *International Journal of Engineering Education*, vol. 30, no. No. 4, 2014.
- [77] K. Verbert and E. Duval, Towards a global architecture for learning objects:, Lugano, Switzerland: The 16th ED-MEDIA 2004World Conference on Educational Multimedia, Hypermedia and Telecommunications, 2004.
- [78] E. Duval, W. Hodgins, D. Rehak and R. Robson, Learning objects symposium special issue guest editorial, *Journal of Educational Multimedia and Hypermedia*, 2004.
- [79] V. Štuikys and R. Damaševičius, Towards knowledge-based generative learning objects, *Information, Technology and Control*, 2007.
- [80] D. Lint, L. Maranda, B. Alina, N. Sorin, Z. Dan and Z. Maria, *Matematica*, manual pentru clasa a opta, Bucharest: Ed. Litera, 2020.
- [81] I. Cicu, D. Eliza, I. Ioana și C. Razvan, *Matematica*, clasa a VIII-a, Bucharest: Ed. Intuitext, 2020.
- [82] Ilias, Open Source e-Learning, *ilias.de*, 2017.
- [83] F. M. Costea, B. C. Chirila and V. I. Cretu, "Designing E-Learning Content Using AGLOs," in *The 23rd International Conference on System Theory, Control and Computing*, Timisoara, Romania, 2019.
- [84] C. B. Chirila, "Auto-Generative Learning Objects for IT Disciplines," in *Proceedings of the International Conference on Virtual Learning*, Bucharest, Romania, 2015.
- [85] C. B. Chirila and P. Gaultier, "Metamodels for Auto-Generative Learning Objects Dedicated to Unix Operating System Disciplines," in *International Conference on System Theory, Control and Computing*, Sinaia, Romania, 2018.
- [86] C. Holotescu, G. Grosseck, D. Andone, L. Gunesch, L. Constandache, V. D. Nedelcu, M. Ivanova and R. Dumbraveanu, "Romanian Educational System Response during the Covid-19 Pandemic," in *Proceedings of the 16th International Scientific Conference "eLearning and Software for Education"* , Bucharest, doi: 10.12753/2066-026X-20-171, 2020, pp. 11 - 20.
- [87] DCMI, "DCMI Grammatical Principles,," Available: www.dublincore.org/usage/documents/principles/. [Accessed: 15-07-2018], 2018.
- [88] Moodle Coordinate Question Plugin, "Tutorial and Documentation," <https://code.google.com/p/moodle-coordinate-question/>, 2015.
- [89] C.-B. Chirila, "Towards the enhancement of AGLOs with SCORM and xAPI," in *The 13th International Scientific Conference eLearning and software for Education*, Bucharest, 2017.

[90] D. Hillmann, "Using Dublin Core," in <http://dublincore.org/documents/usageguide/>, 2003.