ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΚΟΙΝΩΝΙΚΩΝ ΚΑΙ ΠΟΛΙΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΚΟΙΝΩΝΙΚΗΣ ΚΑΙ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

# Ανάπτυξη Εκπαιδευτικού Περιβάλλοντος Δυναμικής Οπτικοποίησης Αλγορίθμων Ταξινόμησης Πινάκων και Μελέτη της Συμβολής του στην Αλγοριθμική Σκέψη Μαθητών Λυκείου

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

του

**ΕΥΡΙΠΙΔΗ ΒΡΑΧΝΟΥ**

Κόρινθος, Ιούλιος 2018

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΚΟΙΝΩΝΙΚΩΝ ΚΑΙ ΠΟΛΙΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΚΟΙΝΩΝΙΚΗΣ ΚΑΙ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

**Τριμελής Συμβουλευτική Επιτροπή:**

Αθανάσιος Τζιμογιάννης, Καθηγητής Πανεπιστημίου Πελοποννήσου (επιβλέπων)

Βασίλειος Κόμης, Καθηγητής Πανεπιστημίου Πατρών

Αναστάσιος Μικρόπουλος, Καθηγητής Πανεπιστημίου Ιωαννίνων

**Επταμελής Εξεταστική Επιτροπή:**

Αθανάσιος Τζιμογιάννης, Καθηγητής Πανεπιστημίου Πελοποννήσου

Βασίλειος Κόμης, Καθηγητής Πανεπιστημίου Πατρών

Αναστάσιος Μικρόπουλος, Καθηγητής Πανεπιστημίου Ιωαννίνων

Χρήστος Κακλαμάνης, Καθηγητής Πανεπιστημίου Πατρών

Γεώργιος Λέπουρας, Καθηγητής Πανεπιστημίου Πελοποννήσου

Χαράλαμπος Καραγιαννίδης, Καθηγητής Πανεπιστημίου Θεσσαλίας

Χρήστος Παναγιωτακόπουλος, Καθηγητής Πανεπιστημίου Πατρών

# Ευχαριστίες

Με την ολοκλήρωση αυτής της διατριβής θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Αθανάσιο Τζιμογιάννη για την ενθάρρυνση, την πολύπλευρη συνεισφορά και την καθοδήγησή του σε όλη τη διάρκεια εκπόνησης της διατριβής. Η συμβολή του σε όλα τα στάδια της διατριβής ήταν καθοριστική, ειδικά όταν με βοήθησε να διαχωρίσω το ρόλο του ερευνητή από αυτόν του εκπαιδευτικού.

Επίσης θα ήθελα να ευχαριστήσω θερμά τα μέλη της τριμελούς επιτροπής παρακολούθησης της διατριβής: τον κύριο Βασίλειο Κόμη, Καθηγητή Πανεπιστημίου Πατρών και τον κύριο Αναστάσιο Μικρόπουλο, Καθηγητή Πανεπιστημίου Ιωαννίνων, που με τίμησαν με την υποστήριξή τους στη δουλειά μου.

Τέλος θα ήθελα να ευχαριστήσω τους μαθητές των σχολείων: Πειραματικό (2ο) Λύκειο Ιλίου, Πειραματικό Λύκειο Αγίων Αναργύρων, 1ο Λύκειο Πετρούπολης, 7ο Λύκειο Περιστερίου, 10ο Λύκειο Περιστερίου, Ζάννειο Πειραματικό Γυμνάσιο και Λύκειο όπως και τους φοιτητές του πρώτου έτους του τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Πελοποννήσου και τον κύριο Λέπουρα που δέχθηκαν να απαντήσουν στα ερωτηματολόγια των ερευνών.

# Περίληψη

*Η διδασκαλία των βασικών αλγοριθμικών δομών σε μαθητές και φοιτητές αποτελεί ένα ανοικτό ερευνητικό πρόβλημα, το οποίο αποκτά συνεχώς ενδιαφέρον με την εξέλιξη των γλωσσών προγραμματισμού και των προγραμματιστικών εργαλείων. Οι πίνακες και οι αλγόριθμοι ταξινόμησης και αναζήτησης είναι από τις πιο προχωρημένες έννοιες που συναντούν οι μαθητές σε ένα εισαγωγικό μάθημα αλγοριθμικής και προγραμματισμού.*

*Οι έννοιες αυτές είναι δύσκολο να γίνουν κατανοητές από τους μαθητές επειδή αναφέρονται σε αφηρημένα αντικείμενα, για τα οποία δεν είναι εύκολο να βρεθούν επιστημονικά συνεπείς και διδακτικά επεξηγηματικές αναπαραστάσεις στον πραγματικό κόσμο και δεν σχετίζονται πάντα με τις προϋπάρχουσες γνώσεις ή τις καθημερινές εμπειρίες των μαθητών. Επίσης, η διδασκαλία τους με συμβατικά μέσα αποτελεί ένα πολύ δύσκολο εγχείρημα λόγω της δυναμικής φύσης τους, η οποία δεν επιτρέπει την γραφική απεικόνιση της λειτουργίας τους με χαρτί και μολύβι.*

*Τα τελευταία χρόνια έχουν αναπτυχθεί διάφορα εκπαιδευτικά περιβάλλοντα οπτικοποίησης αλγορίθμων, τα οποία έχουν ως στόχο να βοηθήσουν τους μαθητές να οικοδομήσουν επαρκείς αναπαραστάσεις για διάφορες προγραμματιστικές έννοιες μέσα από την ανάδειξη σημαντικών χαρακτηριστικών των αλγόριθμων. Η συνεισφορά της διατριβής είναι διττή. Αρχικά διερευνήθηκαν οι προϋπάρχουσες γνώσεις, οι αναπαραστάσεις, οι δυσκολίες και οι παρανοήσεις που έχουν οι μαθητές για τη δομή του πίνακα και τους αλγόριθμους ταξινόμησης. Η ανάλυση των αποτελεσμάτων έγινε με χρήση της ταξινομίας γνωστικών στόχων SOLO.*

*Αξιοποιώντας τα αποτελέσματα της βιβλιογραφικής επισκόπησης και των εμπειρικών ερευνών της διατριβής σχετικά με τις δυσκολίες των μαθητών για την έννοια του πίνακα και τους αλγόριθμους ταξινόμησης πινάκων, καθορίστηκαν τα βασικά χαρακτηριστικά σχεδιασμού ενός νέου εκπαιδευτικού λογισμικού οπτικοποίησης αλγορίθμων. Με βάση αυτά, σχεδιάστηκε και αναπτύχθηκε το διαδικτυακό σύστημα οπτικοποίησης αλγορίθμων DAVE (**D**ynamic **A**lgorithm **V**isualization **E**nvironment), το οποίο μπορεί να εκτελεστεί μέσω φυλλομετρητή σε οποιαδήποτε πλατφόρμα ή συσκευή που έχει πρόσβαση στο Διαδίκτυο.*

*Η εμπειρική μελέτη που ακολούθησε έδειξε ότι το περιβάλλον DAVE ενίσχυσε την προσπάθεια των μαθητών στην επίλυση αλγοριθμικών προβλημάτων με πίνακες, προωθώντας την ενεργό συμμετοχή και τον πειραματισμό τους με οπτικοποιήσεις αλγορίθμων. Οι μαθητές αξιοποίησαν τις δυνατότητες του λογισμικού και, κυρίως, την τροποποίηση του κώδικα του αλγορίθμου και τη βηματική εκτέλεση της οπτικοποίησης. Τα αποτελέσματα έδειξαν ότι τα δυναμικά χαρακτηριστικά του DAVE και οι δυνατότητες εντοπισμού λογικών λαθών συνέβαλαν στην ανάδειξη και διόρθωση λογικών σφαλμάτων και στην οικοδόμηση επαρκών αναπαραστάσεων των μαθητών για τις έννοιες του πίνακα, του δείκτη, της αντιμετάθεσης και της σύγκρισης στοιχείων.*

**Λέξεις Κλειδιά:** Οπτικοποίηση Αλγορίθμων, Πίνακες, Αλγόριθμοι Ταξινόμησης, παρανοήσεις μαθητών στον προγραμματισμό

# Abstract

*The teaching of basic algorithmic structures to students still constitutes an open research problem. Developing students' computational thinking is currently a major objective of primary and secondary education in many countries around the globe. Literature suggests that arrays and sorting algorithms are among the most advanced concepts students encountered in an introductory algorithmic and programming course.*

*These concepts are difficult to comprehend by students because they refer to abstract entities for which it is not easy to find scientifically coherent and didactic real world's explanatory representations that are related to students' pre-existing knowledge or experience. They are also difficult to illustrate by conventional means due to their abstract and dynamic nature. In recent years, various algorithm visualization systems that illustrate the behaviour of algorithms on data structures are proposed as alternative and efficient instructional environments.*

*The contribution of this dissertation is twofold. Firstly, we analysed students' preconceptions, misconceptions about arrays and and their difficulties to solve problems with sorting algorithms. Two empirical studies were conducted: the first concerned secondary education students' misconceptions and mental representations of the array data structure, and the second the difficulties the students faced in understanding sorting algorithms and applying them to solve problems.*

*Based on the results of these empirical studies, we have developed DAVE (Dynamic Algorithm Visualization Environment), a web-based algorithm visualization environment that facilitates students' experimentation with array algorithms by allowing the modification of both code and data. The empirical study that followed showed that DAVE has helped students in solving algorithmic problems by promoting active participation and experimentation with the visualization of the algorithm. The results showed that DAVE's dynamic features and logical error detection helped students to identify and correct logical errors and to build adequate representations about array concepts and sorting algorithms.*



**Keywords**: Algorithm Animation, Arrays, Sorting Algorithms, Visualization, students' misconceptions about programming

## Κατάλογος Σχημάτων

## Κατάλογος Πινάκων

## Κατάλογος Αλγορίθμων

## Κατάλογος Τμημάτων Κώδικα

# Περιεχόμενα

# 1

# *Εισαγωγή*

## *1.1 Αντικείμενο της Διατριβής*

Η ραγδαία εξέλιξη της επιστήμης της πληροφορικής έχει επηρεάσει σημαντικά πολλούς τομείς της ζωής του ανθρώπου. Η έκρηξη της ανάπτυξης εφαρμογών για κινητά τηλέφωνα σε συνδυασμό με την ανάπτυξη του παγκόσμιου ιστού και των κοινωνικών μέσων δικτύωσης έχει οικονομικές, κοινωνικές και πολιτισμικές επιπτώσεις στην καθημερινή ζωή. Η ευκολία με την οποία μπορεί να αναπτυχθεί μια εφαρμογή συλλογής, επεξεργασίας και ανάλυσης δεδομένων από τον παγκόσμιο ιστό και τα κοινωνικά μέσα δικτύωσης καθιστά απαραίτητη την εκμάθηση του προγραμματισμού από τους επιστήμονες και τους μηχανικούς που ασχολούνται με την οργάνωση και ανάλυση δεδομένων. Επίσης οι δεξιότητες επίλυσης προβλήματος που αναπτύσσονται κατά τη σχεδίαση αλγορίθμων και τον προγραμματισμό τους, θεωρούνται εξαιρετικά σημαντικές και δεν εμφανίζονται σε άλλα γνωστικά αντικείμενα της δευτεροβάθμιας εκπαίδευσης (Κόμης & Τζιμογιάννης, 2006· Κόμης, 2005). Οι δεξιότητες αυτές περιγράφονται σήμερα με τον ευρύτερο όρο της υπολογιστικής σκέψης (Wing, 2006).

Για αυτό πολλές πανεπιστημιακές σχολές έχουν συμπεριλάβει μαθήματα προγραμματισμού στα προγράμματα σπουδών τους τα τελευταία χρόνια (Aleksic & Ivanovic, 2016). Επιπρόσθετα πολλές χώρες έχουν εισάγει τον προγραμματισμό υπολογιστών ως ξεχωριστό γνωστικό αντικείμενο στην δευτεροβάθμια αλλά και στην πρωτοβάθμια εκπαίδευση (Balanskat & Engelhardt, 2014). Σε άλλες περιπτώσεις λειτουργούν όμιλοι προγραμματισμού (code clubs) σε πολλά σχολεία στην Ευρώπη και στην Αμερική (Smith, Sutcliffe & Sandvik, 2014).

Γενικότερα υπάρχει μια ριζική αναδόμηση των προγραμμάτων σπουδών της πληροφορικής σε όλες τις βαθμίδες της εκπαίδευσης, όπου το ενδιαφέρον μετατοπίζεται σε έννοιες υπολογιστικής σκέψης και προγραμματισμού υπολογιστών, όπως συμβαίνει με τα Exploring Computer Science (Goode & Margolis, 2011· ECS, 2017) και Computer Science Principles (Astrachan, & Briggs, 2012· Garcia et al., 2015, College Board, 2017) στην Αμερική ή την πρωτοβουλία Computing at Schools στη Μεγάλη Βρετανία (Brown et al., 2013). Παρόλα αυτά τα εισαγωγικά μαθήματα προγραμματισμού και αλγορίθμων συνιστούν, ακόμη και σήμερα, ένα πολύ δύσκολο αντικείμενο για τους μαθητές και τους καθηγητές (Kunkle & Allen, 2016· de Raadt, 2007· Robins, Rountree, & Rountree, 2003· Milne & Rowe, 2002· McCracken et al., 2001). Πολλοί ερευνητές δέχονται ότι υπάρχει σημαντικό ποσοστό φοιτητών που δεν καταφέρνουν να ολοκληρώσουν επιτυχώς ένα εισαγωγικό μάθημα προγραμματισμού (Pappas, Giannakos & Jaccheri, 2016· Petersen et al., 2016· Watson & Frederick, 2014· Bennedsen & Caspersen, 2007).

Σύμφωνα με την βιβλιογραφία οι μαθητές αντιμετωπίζουν μεγάλες δυσκολίες σε σύνθετες έννοιες όπως είναι οι δομές δεδομένων και οι αλγόριθμοι (Karpierz & Wolfman, 2014· Danielsiek, Wolfgang & Vahrenhold, 2012· Lahtinen, Ala-Mutka & Järvinen, 2005). Αυτό έπρεπε να είναι αναμενόμενο αφού οι αλγόριθμοι αυτοί οικοδομούνται πάνω στις έννοιες του πίνακα και της δομής επανάληψης που ήδη θεωρούνται οι δύο πιο δύσκολες έννοιες για τους μαθητές με βάση την εμπειρία, αλλά και διάφορες έρευνες που έχουν γίνει σε εκπαιδευτικούς πληροφορικής (Dale, 2006).

Οι έννοιες αυτές είναι δύσκολο να γίνουν κατανοητές από τους μαθητές επειδή αναφέρονται σε αφηρημένα αντικείμενα για τα οποία δεν είναι εύκολο να βρεθούν επιστημονικά συνεπείς και διδακτικά επεξηγηματικές αναπαραστάσεις του φυσικού κόσμου. Επίσης, η διδασκαλία τους με συμβατικά μέσα αποτελεί ένα πολύ δύσκολο εγχείρημα λόγω της δυναμικής τους φύσης, η οποία δεν επιτρέπει τη γραφική απεικόνιση της λειτουργίας τους με χαρτί και μολύβι (Cetin & Andrews-Larson, 2016· Boticki et al., 2013).

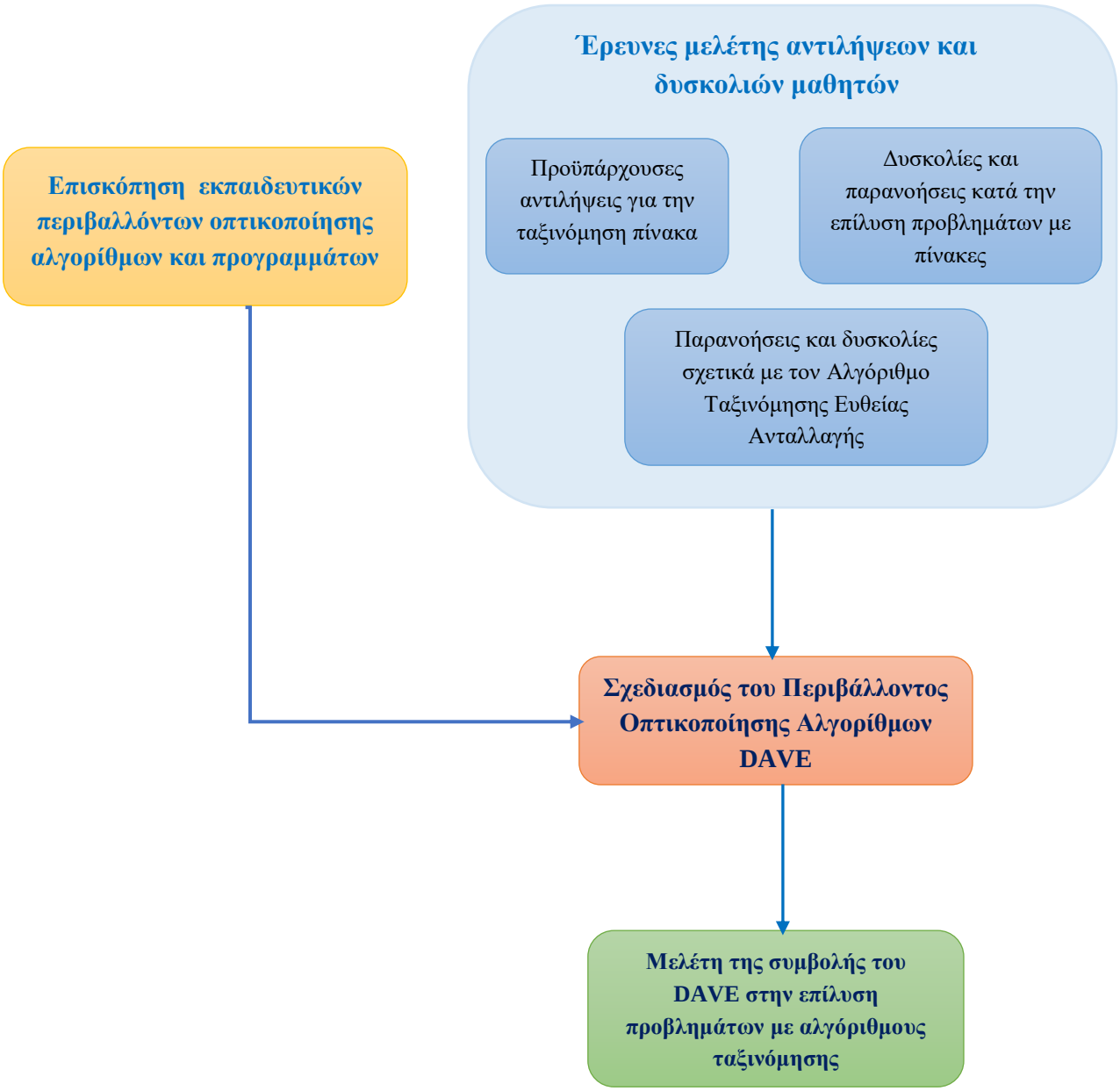
Σύμφωνα με τους Κόμη & Τζιμογιάννη (2006), τα συνήθη προγραμματιστικά περιβάλλοντα και οι γλώσσες προγραμματισμού που χρησιμοποιούνται, έχουν σχεδιαστεί για την ανάπτυξη εφαρμογών και όχι για τη διδασκαλία του προγραμματισμού. Είναι, συνεπώς, προσαρμοσμένα στο πλαίσιο γνώσεων και

δεξιοτήτων των έμπειρων προγραμματιστών, γεγονός που ενισχύει τις δυσκολίες και τα εμπόδια που συναντούν οι μαθητές και οι αρχάριοι στον προγραμματισμό.

Για την αντιμετώπιση των δυσκολιών των αρχάριων προγραμματιστών, έχουν προταθεί κατάλληλα σχεδιασμένες μαθησιακές δραστηριότητες με χρήση εκπαιδευτικών περιβαλλόντων προσομοίωσης-οπτικοποίησης δομών δεδομένων και αλγορίθμων (Halim, 2015· Hundhausen & Brown, 2007· Karavirta & Malmi, 2013· Sorva, Urquiza–Fuentes & Velazquez–Iturbide, 2009). Τα περιβάλλοντα αυτά παρέχουν νέες δυνατότητες για την οικοδόμηση γνώσεων και την ανάπτυξη δεξιοτήτων στον προγραμματισμό. Το βασικό χαρακτηριστικό τους είναι η αναπαράσταση της δυναμικής φύσης των αφηρημένων δομών και αλγορίθμων που είναι δύσκολο να αναπαρασταθούν με συμβατικά μέσα. Κύριοι άξονες των εποικοδομητικών διδακτικών προσεγγίσεων είναι η εκτίμηση των προϋπαρχουσών γνώσεων και αντιλήψεων των μαθητών και η οργάνωση διδακτικών-μαθησιακών δραστηριοτήτων που να ευνοούν τη διερευνητική, ανακαλυπτική και συνεργατική μάθηση. Οι προσεγγίσεις αυτές δίνουν έμφαση στον παιδαγωγικό σχεδιασμό της διδασκαλίας του προγραμματισμού και στη μετατόπιση από το συντακτικό στην καλλιέργεια δεξιοτήτων επίλυσης προβλημάτων (αναλυτική-συνθετική σκέψη, αφαιρετική ικανότητα, μοντελοποίηση λύσεων) (Τζιμογιάννης, 2005).

## *1.2 Σχεδιασμός και συμβολή της διατριβής*

Στο Σχήμα 1.1 παρουσιάζεται ο σχεδιασμός και οι φάσεις υλοποίησης της παρούσας διδακτορικής διατριβής. Η βιβλιογραφική επισκόπηση έδειξε ότι υπάρχει μεγάλο έλλειμμα ερευνητικών εργασιών για τη διερεύνηση των δυσκολιών που αντιμετωπίζουν οι μαθητές στους πίνακες και στους αλγόριθμους ταξινόμησης. Παρόλο που ο πίνακας αποτελεί μια θεμελιώδη δομή δεδομένων και μάλιστα στα περισσότερα εισαγωγικά μαθήματα προγραμματισμού είναι η πρώτη δομή δεδομένων με την οποία έρχονται σε επαφή οι μαθητές, από την αναζήτηση της βιβλιογραφίας δεν βρέθηκε καμία εμπειρική έρευνα για τη μελέτη των δυσκολίων και των παρανοήσεων που αντιμετωπίζουν οι μαθητές με τους πίνακες. Καμία αντίστοιχη έρευνα δεν βρέθηκε επίσης στη βιβλιογραφία για τις δυσκολίες που αντιμετωπίζουν οι μαθητές στους αλγόριθμους ταξινόμησης και στις προϋπάρχουσες γνώσεις που έχουν για αυτούς. Η διατριβή αυτή φιλοδοξεί να συνεισφέρει στη μελέτη του συγκεκριμένου εκπαιδευτικού προβλήματος σχετικά με τη διδασκαλία αλγορίθμων σε πίνακες.




**Σχήμα 1.1. Σχεδιασμός και φάσεις της Διατριβής**

Αρχικά έγινε μελέτη των αναπαραστάσεων των μαθητών για τις έννοιες του πίνακα και των αλγόριθμων ταξινόμησης και παράλληλα εντοπισμός των δυσκολιών και των παρανοήσεων που διατηρούν οι μαθητές για τις έννοιες αυτές.

Για τη μελέτη των προϋπαρχουσών γνώσεων των μαθητών στους αλγόριθμους ταξινόμησης διενεργήθηκαν έρευνες, μια σε μαθητές Λυκείου (Βραχνός & Τζιμογιάννης, 2016) και μια σε μαθητές Γυμνασίου (Βραχνός & Τζιμογιάννης, 2018). Οι έρευνες αυτές έδειξαν ότι οι αλγόριθμοι που χρησιμοποιούν οι μαθητές για να τοποθετήσουν σε αύξουσα διάταξη μια σειρά από αντικείμενα, είναι οι αλγόριθμοι εισαγωγής και επιλογής. Αντίθετα ο αλγόριθμος ταξινόμησης της ευθείας ανταλλαγής

που διδάσκεται στην δευτεροβάθμια εκπαίδευση χρησιμοποιήθηκε από ελάχιστους μαθητές, αφού όπως αποδείχθηκε δεν συνδέεται με τις προϋπάρχουσες γνώσεις και τις καθημερινές εμπειρίες των μαθητών.

Για την μελέτη των αναπαραστάσεων των μαθητών σχετικά με την έννοια του πίνακα και των δυσκολιών που αντιμετωπίζουν κατά την επίλυση προγραμματιστικών προβλημάτων με πίνακες, διενεργήθηκε έρευνα σε 102 μαθητές Γ' τάξης Λυκείου. Η ανάλυση των απαντήσεων των μαθητών έγινε με βάση την ταξινομία SOLO (Vrachnos & Jimoyiannis, 2017). Τα αποτελέσματα έδειξαν ελλιπείς αναπαραστάσεις για την έννοια και τη χρησιμότητα του πίνακα σε αλγόριθμους επεξεργασίας δεδομένων του ίδιου τύπου, καθώς και δυσκολίες στο χειρισμό υπολογιστικών δομών όπου εμφανίζονται σε αλγορίθμους σχετικά με πίνακες. Επίσης διαπιστώθηκε ότι κάποιες παρανοήσεις των μαθητών για την έννοια του πίνακα έχουν ως απαρχή παρανοήσεις για την έννοια της προγραμματιστικής μεταβλητής. Η ερμηνεία για αυτό είναι ότι η μεταβλητή αποτελεί δομικό στοιχείο πάνω στο οποίο οικοδομείται η έννοια του πίνακα, οπότε είναι λογικό οι παρανοήσεις στην έννοια της μεταβλητής να μεταφέρονται ή και να προκαλούν νέες παρανοήσεις στους πίνακες.

Για τη μελέτη των αναπαραστάσεων των μαθητών σχετικά με τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής, διενεργήθηκε μια δεύτερη έρευνα σε δείγμα 136 μαθητών Γ' τάξης τριών γενικών λυκείων της Δυτικής Αττικής και 50 πρωτοετών φοιτητών του Τμήματος Επιστήμης και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πελοποννήσου που κλήθηκαν να απαντήσουν σε αλγοριθμικά προβλήματα ταξινόμησης πινάκων (Βραχνός & Τζιμογιάννης, 2014α). Η ανάλυση των αποτελεσμάτων, που έγινε με βάση την ταξινομία SOLO, έδειξε ότι οι φοιτητές και οι μαθητές είχαν ελλιπείς αναπαραστάσεις για τον αλγόριθμο ταξινόμησης ενώ παρουσίασαν τις ίδιες δυσκολίες στην κατανόηση του αλγόριθμου ταξινόμησης. Οι μαθητές χειρίστηκαν καλύτερα προβλήματα που απαιτούσαν απλά την ανάκληση του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής. Από την άλλη μεριά, οι φοιτητές είχαν καλύτερη απόδοση σε προβλήματα που δεν απαιτούσαν την απλή εφαρμογή ενός αλγορίθμου ταξινόμησης αλλά την επίλυση ενός προβλήματος που είχε σχέση με τον αλγόριθμο ταξινόμησης. Τέλος, έγινε σύνδεση των δυσκολιών που παρουσιάζουν οι μαθητές στον αλγόριθμο ταξινόμησης με τις παρανοήσεις που ήδη έχουν για την έννοια του πίνακα και την έννοια της μεταβλητής.

Με βάση την ανάλυση των αποτελεσμάτων των παραπάνω ερευνών σχεδιάσαμε το σύστημα οπτικοποίησης αλγορίθμων DAVE (**D**ynamic **A**lgorithm **V**isualization **E**nvironment), το οποίο αποτελεί ένα εκπαιδευτικό περιβάλλον που επιτρέπει τη δυναμική οπτικοποίηση αλγορίθμων ταξινόμησης και αναζήτησης σε πίνακες, αναδεικνύοντας τα ιδιαίτερα χαρακτηριστικά (κρίσιμα στοιχεία) της λογικής κάθε αλγορίθμου (Vrachnos & Jimoyiannis, 2014). Χρησιμοποιήθηκε τεχνολογία HTML5/Javascript, η οποία καθιστά το DAVE πλήρως ανεξάρτητο από την πλατφόρμα εκτέλεσης. Συνεπώς, μπορεί να εκτελεστεί σε οποιαδήποτε συσκευή υποστηρίζει πλοήγηση στον Παγκόσμιο Ιστό, όπως είναι τα κινητά και οι ταμπλέτες, με χρήση μόνο ενός φυλλομετρητή (browser).

Το DAVE προωθεί τον πειραματισμό με την οπτικοποίηση του αλγορίθμου και υποστηρίζει τους μαθητές ώστε να οικοδομήσουν αποτελεσματικές αναπαραστάσεις για δομές και αλγορίθμους σε πίνακες. Το σύστημα υποστηρίζει όλους τους γνωστούς αλγόριθμους ταξινόμησης και αναζήτησης που διδάσκονται στην δευτεροβάθμια εκπαίδευση, ενώ επιτρέπει ακόμη και την τροποποίηση του κώδικα των αλγορίθμων σε κάποιες περιπτώσεις, επιτυγχάνοντας τον μέγιστο βαθμό διαδραστικότητας με τον χρήστη. Η δυνατότητα της τροποποίησης του κώδικα του αλγορίθμου, από όσο γνωρίζουμε, προσφέρεται μόνο από το σύστημα PyAlgoViz (Laffra, 2014), το οποίο έχει όμως μια σειρά από μειονεκτήματα σε σχέση με το DAVE, όπως θα αναλύσουμε στη συνέχεια.

Η τελική έρευνα αφορά τη μελέτη της συμβολής του περιβάλλοντος στην κατανόηση αλγορίθμων επεξεργασίας πινάκων (Βραχνός & Τζιμογιάννης, 2014β). Έγινε σε 45 μαθητές Γ' Λυκείου και επικεντρώθηκε στη μελέτη του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής, ευρύτερα γνωστού και ως αλγόριθμου φυσαλίδας. Τα αποτελέσματα έδειξαν ότι οι δυνατότητες πειραματισμού με την οπτικοποίηση της εκτέλεσης του αλγορίθμου, ενθάρρυναν τη συμμετοχή και την ενεργοποίηση των μαθητών κατά την επίλυση προβλημάτων ταξινόμησης πινάκων αυξημένης δυσκολίας. Ειδικά η δυνατότητα τροποποίησης του κώδικα του αλγορίθμου, βοήθησε πολύ τους μαθητές να διερευνήσουν εκτενώς τη συμπεριφορά του αλγόριθμου και να αναπτύξουν τους δικούς τους αλγόριθμους για την επίλυση προβλημάτων ταξινόμησης.

## *1.3 Διάρθρωση της διατριβής*

Η παρούσα διατριβή έχει οργανωθεί σε οκτώ κεφάλαια ως εξής:

Το πρώτο κεφάλαιο περιλαμβάνει μια εισαγωγή στο αντικείμενο, την οριοθέτηση του ερευνητικού προβλήματος, τη συμβολή και τη δομή της διατριβής. Στο δεύτερο κεφάλαιο παρουσιάζεται το θεωρητικό πλαίσιο της διατριβής. Το κεφάλαιο επικεντρώνεται σε βασικά στοιχεία της διδακτικής του προγραμματισμού και σε μια επισκόπηση των δυσκολιών που συναντούν οι μαθητές σε βασικές προγραμματιστικές έννοιες.

Στο τρίτο κεφάλαιο γίνεται μια επισκόπηση των υπαρχόντων εκπαιδευτικών περιβαλλόντων οπτικοποίησης αλγορίθμων και προγραμμάτων. Το κεφάλαιο καταλήγει με τη διατύπωση των επιθυμητών χαρακτηριστικών που πρέπει να έχει ένα σύγχρονο σύστημα οπτικοποίησης αλγορίθμων.

Στο τέταρτο κεφάλαιο παρουσιάζονται τα αποτελέσματα της αρχικής έρευνας για την μελέτη των προϋπαρχουσών γνώσεων των μαθητών στους αλγόριθμους ταξινόμησης. Η έρευνα έγινε σε ένα δείγμα 286 μαθητών Γυμνασίου και Λυκείου και παρουσιάζει τους αλγόριθμους ταξινόμησης που επινοούν οι μαθητές για να τοποθετήσουν σε αύξουσα σειρά μια ομάδα αντικειμένων χωρίς προηγουμένως να έχουν διδαχθεί κάποιον αντίστοιχο αλγόριθμο.

Στα επόμενα δυο κεφάλαια (πέμπτο και έκτο) παρουσιάζονται τα αποτελέσματα δύο ερευνών για τις αναπαραστάσεις και τις δυσκολίες των μαθητών στους πίνακες και στους αλγόριθμους ταξινόμησης, αντίστοιχα. Με βάση τα αποτελέσματα αυτών των ερευνών έγινε ο σχεδιασμός του συστήματος οπτικοποίησης που περιγράφεται στο έκτο κεφάλαιο μαζί με την αρχιτεκτονική και τη λειτουργικότητα του συστήματος.

Στο έβδομο κεφάλαιο παρουσιάζεται ο σχεδιασμός και τα βασικά χαρακτηριστικά του συστήματος. Επίσης παρουσιάζονται όλοι οι αλγόριθμοι ταξινόμησης και αναζήτησης που υποστηρίζει το DAVE.

Στο όγδοο κεφάλαιο παρουσιάζουμε την τελική έρευνα που έγινε για τη μελέτη της συμβολής του συστήματος DAVE σε ένα δείγμα 45 μαθητών δύο γενικών λυκείων. Οι μαθητές κλήθηκαν να σχεδιάσουν αλγορίθμους για την επίλυση προβλημάτων ταξινόμησης με τη βοήθεια του λογισμικού οπτικοποίησης. Τα αποτελέσματα ανέδειξαν σημαντικές πληροφορίες για τη συμβολή του λογισμικού στην εξέλιξη της αλγοριθμικής σκέψης των μαθητών και στην οικοδόμηση αλγορίθμων με στόχο την επίλυση προβλημάτων ταξινόμησης πινάκων.

Στο ένατο κεφάλαιο παρουσιάζονται τα συμπεράσματα της διατριβής, συνοψίζοντας τα επιμέρους συμπεράσματα των ερευνών που αναλύθηκαν στα προηγούμενα κεφάλαια. Γίνεται μια σύγκριση με αντίστοιχες έρευνες και άλλα συστήματα οπτικοποίησης και μια συζήτηση για τους τρόπους αξιοποίησης του λογισμικού στη διδακτική πράξη. Τέλος δίνονται κάποιες κατευθύνσεις για μελλοντική έρευνα και αναφέρονται οι δημοσιεύσεις που στοιχειοθετούν τη συνεισφορά της διατριβής στον χώρο της διδακτικής της πληροφορικής και των συστημάτων οπτικοποίησης αλγορίθμων.

# 2

# *Θεωρητικό Πλαίσιο*

## *2.1 Αλγοριθμική και Προγραμματισμός*

Μια από τις κεντρικές έννοιες της πληροφορικής είναι η έννοια του αλγόριθμου. Ο αλγόριθμος μπορεί να οριστεί ως μια καλά ορισμένη, πεπερασμένη υπολογιστική διαδικασία με στόχο την επίλυση ενός υπολογιστικού προβλήματος. Υπάρχουν διάφοροι αλγόριθμοι που χρησιμοποιούν τα παιδιά από μικρή ηλικία όπως είναι οι αλγόριθμοι του πολλαπλασιασμού, της εύρεσης του μέγιστου κοινού διαιρέτη κ.α. Σε αυτές όμως τις περιπτώσεις οι μαθητές απλά εφαρμόζουν τα βήματα του αλγόριθμου που έχουν διδαχθεί. Η αλγοριθμική έχει ως κεντρική έννοια την επίλυση προβλήματος, τον σχεδιασμό της αλγοριθμικής λύσης και στο τέλος την κωδικοποίηση της λύσης σε κάποια γλώσσα προγραμματισμού, ώστε να μπορεί να εκτελεστεί από ένα υπολογιστικό σύστημα.

Ο προγραμματισμός υπολογιστών θεωρείται πλέον ως μια γνωστική δραστηριότητα η οποία οδηγεί στην ανάπτυξη δεξιοτήτων υψηλού επιπέδου. Η σημασία του προγραμματισμού, ως γνωστική δραστηριότητα των μαθητών, και η συνεισφορά του στην ανάπτυξη δομημένης σκέψης έχει τεθεί για πρώτη φορά από τον Papert (1980).

Η μάθηση του προγραμματισμού δεν αφορά απλά την εκμάθηση μιας γλώσσας προγραμματισμού, αλλά τη οικοδόμηση μιας ικανότητας υψηλού επιπέδου η οποία περιλαμβάνει ένα σύνολο από γνώσεις και δεξιότητες επίλυσης προβλημάτων, που ήταν άγνωστες στους μαθητές πριν την εμφάνιση της πληροφορικής. Οι δεξιότητες αυτές διαφέρουν πολύ από αυτές που συναντώνται στον παραδοσιακό τρόπο επίλυσης προβλημάτων όπου η λύση των προβλημάτων δεν έχει αλγοριθμική μορφή, δηλαδή δεν συνίσταται σε μια ακολουθία βημάτων. Πριν την έλευση της αλγοριθμικής οι σπάνια

είχε ζητηθεί από τους μαθητές να επινοήσουν έναν νέο αλγόριθμο. Μέχρι τότε οι μαθητές εκτελούσαν αλγόριθμους επίλυσης προβλημάτων τους οποίους είχαν διδαχθεί, ανάλογα με το γνωστικό αντικείμενο, όπως είναι για παράδειγμα ο αλγόριθμος εύρεσης του μέγιστου κοινού διαιρέτη, του ελάχιστου κοινού πολλαπλάσιου, ή του υπολογισμού των συντελεστών μιας χημικής αντίδρασης.

Αντίθετα, κατά την επίλυση προβλημάτων σε προγραμματιστικό περιβάλλον, χρησιμοποιούνται βασικές έννοιες όπως μεταβλητή, δομές επανάληψης, διαδικασίες αλλά και πιο προχωρημένες όπως η αναδρομή που απαιτούν υψηλού επιπέδου ικανότητα γενίκευσης από τους μαθητές, για τη διατύπωση του γενικού αναδρομικού βήματος. Αυτές οι έννοιες είναι πολύ δύσκολο να οικοδομηθούν στο πλαίσιο άλλων διδακτικών αντικειμένων στο σχολείο. Αυτό το γεγονός είναι που καθιστά επιστημονικά έγκυρη την ένταξη του προγραμματισμού και της αλγοριθμικής στο αναλυτικό πρόγραμμα του σχολείου.

## *2.2 Ο προγραμματισμός στα αναλυτικά προγράμματα σπουδών*

Η εισαγωγή της πληροφορικής ως αυτόνομου γνωστικού αντικειμένου στο πρόγραμμα σπουδών ξεκίνησε στις αρχές της δεκαετίας του 1970. Παράλληλα με τη σταδιακή αυτονόμηση της πληροφορικής και τη δημιουργία αμιγών πανεπιστημιακών τμημάτων, σταδιακά πολλές ανεπτυγμένες χώρες εντάσσουν στο πρόγραμμα σπουδών του Λυκείου μαθήματα πληροφορικής όπως η Γαλλία. Σύμφωνα με τον Κόμη (2005) την περίοδο αυτή η πληροφορική ταυτίζεται σε μεγάλο βαθμό με τον προγραμματισμό. Ενώ τη δεκαετία του 1980 υπάρχει μια τάση εισαγωγής της πληροφορικής σε όλες τις βαθμίδες, υιοθετώντας μια σφαιρική προσέγγιση, τη δεκαετία του 1990 η τάση αυτή ανακόπτεται και παρατηρείται υποβάθμιση της πληροφορικής ως επιστήμης. Παράλληλα, η ραγδαία ανάπτυξη και διάδοση της χρήσης των προσωπικών υπολογιστών οδήγησε στην ενσωμάτωση των ΤΠΕ σε άλλα μαθήματα.

Την τελευταία δεκαετία έχουν αναπτυχθεί αρκετά νέα προγράμματα σπουδών για το γνωστικό αντικείμενο της επιστήμης της πληροφορικής σε διάφορες χώρες, όπου παρατηρείται μια μετατόπιση από τις Τεχνολογίες Πληροφορικής και Επικοινωνιών (ΤΠΕ) στο γνωστικό αντικείμενο της επιστήμης της πληροφορικής (Bell, Andreae & Lambert, 2010· Brown et al., 2013· Hubwieser, 2012) και κυρίως στην σχεδίαση αλγορίθμων και στον προγραμματισμό.

Ωστόσο υπήρχαν χώρες που προσέφεραν προχωρημένα μαθήματα πληροφορικής στα σχολεία αρκετά νωρίτερα όπως το Ισραήλ (Gal-Ezer & Stephenson, 2014· Gal-Ezer et al., 1995) και πολλές ανατολικοευρωπαϊκές χώρες (Dagienė, 2008· Sysło, 2011). Το πρόγραμμα σπουδών του Ισραήλ (Gal-Ezer et al., 1995) ήταν ιδιαίτερα απαιτητικό αφού περιλάμβανε πολύ προχωρημένα αντικείμενα όπως μεταγλωττιστές, πολυπλοκότητα αλγορίθμων και τεχνολογία λογισμικού. Η επιστημονική επιτροπή με επικεφαλής την Gal-Ezer δεσμεύτηκε να τοποθετήσει την επιστήμη των υπολογιστών στο ίδιο επίπεδο με τη φυσική, τη χημεία και τη βιολογία, έχοντας στο επίκεντρο την έννοια της υπολογιστικής σκέψης.

Μέχρι σήμερα, πολυάριθμες πρωτοβουλίες και έργα έχουν ξεκινήσει για την προώθηση της πληροφορικής στην δευτεροβάθμια εκπαίδευση. Ένα από τα πιο σύγχρονα και πιο δημοφιλή προγράμματα σπουδών που αποτελεί σημείο αναφοράς είναι το προτεινόμενο πρόγραμμα του Association for Computing Machinery (ACM) (Tucker et al., 2006). Όταν η ACM δημιούργησε την Ένωση Καθηγητών Πληροφορικής (**C**omputer **S**cience **T**eachers **A**ssociation) το 2004, η CSTA ανέλαβε καθήκοντα συνεχούς αναθεώρησης και ενημέρωσης του προγράμματος και έτσι καταλήξαμε στην πιο πρόσφατη έκδοσή του (Seehorn et al., 2011). Η θεματολογία και οι διδακτικοί στόχοι εκτείνονται από τις τάξεις του δημοτικού μέχρι τις τάξεις του Λυκείου με μερικά εξειδικευμένα μαθήματα επιλογής. Σε αυτό το πρόγραμμα σπουδών δίνεται βάρος στον προγραμματισμό τη σχεδίαση αλγορίθμων και την υπολογιστική σκέψη. Μια άλλη δράση είναι η "Πληροφορική για όλους" για την Ευρώπη όπου πάλι συνεργάζεται η ACM με τον ευρωπαϊκό φορέα Informatics Europe (Caspersen et al., 2018).

Στις ΗΠΑ το εκπαιδευτικό σύστημα είναι αποκεντρωμένο και κάθε πολιτεία σχεδιάζει το δικό της πρόγραμμα σπουδών. Μάλιστα σε κάποιες περιπτώσεις υπάρχουν διαφορές ακόμα και μεταξύ εκπαιδευτικών περιφερειών της ίδιας πολιτείας. Για αυτό το λόγο τα τελευταία χρόνια έχουν σχεδιαστεί διάφορα προγράμματα σπουδών για την πληροφορική όπως το Exploring Computer Science (ECS, 2017· Goode & Margolis, 2011) που έγινε αρχικά για τα σχολεία του Λος Άντζελες. Το πρόγραμμα είναι ενός έτους και χωρίζεται σε έξι θεματικές ενότητες (επίλυση προβλήματος, προγραμματισμός, ανάλυση δεδομένων, ρομποτική, επικοινωνία ανθρώπου-μηχανής, σχεδίαση ιστοσελίδων). Περιέχει αρκετό υλικό, αναλυτικά σχέδια μαθήματος και

φύλλα εργασίας και ανανεώνεται και εμπλουτίζεται συνεχώς. Στο πρόγραμμα δίνεται έμφαση στην υπολογιστική σκέψη και τον προγραμματισμό.

Παράλληλα με το ECS σχεδιάστηκε το Computer Science Principles (Astrachan, & Briggs, 2012· College Board, 2017· Garcia et al., 2015) το οποίο έχει στόχο την προετοιμασία των υποψηφίων για τις εισαγωγικές εξετάσεις (Advanced Placement Program) στα ΑΕΙ. Το πρόγραμμα δεν δίνει έμφαση μόνο στον προγραμματισμό και την υπολογιστική σκέψη, αλλά περιλαμβάνει και άλλες περιοχές της επιστήμης της πληροφορικής.

Ταυτόχρονα, η Βρετανική Βασιλική Εταιρεία δημοσίευσε το γνωστό "Shutdown or Restart" (Furber, 2012) για τη ριζική αναμόρφωση της πληροφορικής στην πρωτοβάθμια και δευτεροβάθμια εκπαίδευση. Σύμφωνα με αυτή την έκθεση ο όρος ΤΠΕ (ICT) παύει να χρησιμοποιείται και αντικαθίσταται από τον όρο Υπολογιστική (Computing) με ταυτόχρονη μετατόπιση του αντικείμενου στον προγραμματισμό υπολογιστών. Η έκθεση αυτή ενέπνευσε την ευρεία και πολλά υποσχόμενη πρωτοβουλία "Computing at Schools" (Brown et al., 2013) για την εισαγωγή της πληροφορικής ως γνωστικό αντικείμενο στα σχολεία. Στο πρόγραμμα αυτό τα παιδιά εισάγονται στον προγραμματισμό από το δημοτικό ενώ η υπολογιστική σκέψη έχει πάλι δεσπόζουσα θέση. Τέσσερα χρόνια μετά η Βασιλική Ακαδημία (2017) δημοσίευσε μια αποτίμηση της προσπάθειας αυτής με τίτλο "After the reboot: computing education in UK schools" στην οποία τονίζει την ανάγκη για επιμόρφωση των καθηγητών πληροφορικής ώστε να μπορούν να ανταπεξέλθουν στις προκλήσεις του νέου προγράμματος σπουδών.

Είναι φανερό λοιπόν ότι τα τελευταία χρόνια υπάρχει μια τάση εισαγωγής της πληροφορικής ως επιστημονικό αντικείμενο στην δευτεροβάθμια εκπαίδευση σε πολλές χώρες του κόσμου με επίκεντρο την υπολογιστική σκέψη, τη σχεδίαση αλγορίθμων και τον προγραμματισμό υπολογιστών.

## *2.3 Ο προγραμματισμός στο ελληνικό εκπαιδευτικό σύστημα*

Στην ελληνική εκπαίδευση ο προγραμματισμός ξεκίνησε ως μάθημα ειδίκευσης το 1985 στα Τεχνικά Επαγγελματικά Λύκεια (ΤΕΛ) με την Basic και την Pascal και στα Ενιαία Πολυκλαδικά Λύκεια (ΕΠΛ) με την Cobol. Τη δεκαετία του 1990 ο προγραμματισμός αποτέλεσε ενότητα του μαθήματος της πληροφορικής που εισήχθη ως αυτόνομο γνωστικό αντικείμενο πιλοτικά σε κάποια Γυμνάσια και Λύκεια της

χώρας. Στα γυμνάσια επιλέχθηκε η γλώσσα Logo την Α' τάξη και η γλώσσα Basic για την Γ' τάξη. Η τεχνοκεντρική προσέγγιση (Τζιμογιάννης, 2003) που ακολουθήθηκε έδινε βάρος στην εκμάθηση των χαρακτηριστικών συγκεκριμένων γλωσσών προγραμματισμού όπως κανόνες σύνταξης, δομή κλπ.

Από την άλλη, η αποδέσμευση της διδασκαλίας από τις τεχνικές λεπτομέρειες μιας γλώσσας προγραμματισμού και η μετατόπιση σε δεξιότητες επίλυσης προβλήματος, ακολουθήθηκε με το Ενιαίο Πλαίσιο Προγράμματος Σπουδών Πληροφορικής (ΥΠΕΠΘ, 1998). Στο πλαίσιο αυτό καθιερώνεται στην τεχνολογική κατεύθυνση της Γ' τάξης του Λυκείου το μάθημα με τίτλο "Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον" που έχει στόχο την καλλιέργεια δεξιοτήτων επίλυσης αλγοριθμικών προβλημάτων. Το μάθημα διδάσκεται σήμερα 2 ώρες την εβδομάδα στους μαθητές της ομάδας προσανατολισμού οικονομίας και πληροφορικής ως πανελλαδικά εξεταζόμενο και στους μαθητές της ομάδας προσανατολισμού θετικών σπουδών ως μάθημα κατεύθυνσης. Εκτός από τις βασικές αλγοριθμικές οι μαθητές εισάγονται στη δομή δεδομένων του πίνακα και στους αλγόριθμους της σειριακής και δυαδικής αναζήτησης και τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής.

Επίσης στην Α' τάξη του Γενικού Λυκείου διδάσκεται το μάθημα επιλογής "Εφαρμογές Πληροφορικής" ενώ στην Β' τάξη το μάθημα γενικής παιδείας "Εισαγωγή στις αρχές της επιστήμης των Η/Υ". Στο μάθημα επιλογής χρησιμοποιούνται τα οπτικά περιβάλλοντα προγραμματισμού Alice και App Inventor για την ανάπτυξη απλών εφαρμογών. Στο μάθημα της Β' λυκείου οι μαθητές έχουν την πρώτη τους επαφή με μια μη οπτική γλώσσα στην οποία πρέπει να γράψουν κώδικα, την ψευδογλώσσα η οποία ωστόσο δεν αποτελεί μια συντακτικά αυστηρή γλώσσα προγραμματισμού.

Τέλος στο επαγγελματικό λύκειο το νέο πρόγραμμα σπουδών (ΦΕΚ 2010/Τβ/16-09-2015) περιλαμβάνει τρία μαθήματα προγραμματισμού στις ειδικότητες του τομέα πληροφορικής. Στη Β' λυκείου το μάθημα Αρχές Προγραμματισμού εισάγει τους μαθητές στον προγραμματισμό με τη γλώσσα Python. Η τελευταία ενότητα του μαθήματος περιλαμβάνει και την παρουσίαση του αλγορίθμου της σειριακής αναζήτησης και του αλγορίθμου ταξινόμησης με επιλογή. Στην τρίτη τάξη το μάθημα "Προγραμματισμός Υπολογιστών" είναι συνέχεια του μαθήματος της Β' τάξης. Στο μάθημα χρησιμοποιείται πάλι η γλώσσα Python και καλύπτονται πιο προχωρημένες έννοιες όπως δομές δεδομένων, αντικειμενοστρεφής προγραμματισμός και αρχεία. Το μάθημα έχει ξεχωριστή ενότητα όπου παρουσιάζονται οι αλγόριθμοι σειριακής και

δυαδικής αναζήτησης και οι αλγόριθμοι ταξινόμησης ευθείας ανταλλαγής, εισαγωγής και επιλογής. Το μάθημα αυτό εξετάζεται πανελλαδικά ως μάθημα ειδικότητας για την εισαγωγή των μαθητών του τομέα πληροφορικής των επαγγελματικών λυκείων στην τριτοβάθμια εκπαίδευση. Στην τρίτη τάξη του επαγγελματικού λυκείου υπάρχει ένα ακόμα μάθημα προγραμματισμού με τίτλο " Ειδικά θέματα στον Προγραμματισμό Υπολογιστών" το οποίο εισάγει τους μαθητές στον αντικειμενοστρεφή προγραμματισμό με τη γλώσσα Java.

Στο δημοτικό έχει εισαχθεί από το 2011 το μάθημα Τεχνολογίες Πληροφορικής και Επικοινωνιών (ΥΠΔΒΜΘ, 2011α) που εισάγει τον όρο του πληροφορικού γραμματισμού (ICT literacy) για να περιγράψει την ικανότητα των μαθητών να χρησιμοποιούν τις σύγχρονες ψηφιακές τεχνολογίες για την προσπέλαση, διαχείριση, αξιολόγηση και δημιουργία πληροφοριών, με στόχο την επίλυση προβλημάτων. Στην προγραμματισμού του οι μαθητές εμπλέκονται σε δραστηριότητες επίλυσης προβλημάτων που έχουν ως σκοπό την καλλιέργεια δεξιοτήτων μεθοδολογικού χαρακτήρα (επεξεργασία δεδομένων, σχεδιασμός και υλοποίηση αλγορίθμων, μοντελοποίηση λύσεων, δημιουργικότητα και καινοτομία) και δεξιοτήτων υψηλού επιπέδου (διερεύνηση, κριτική και αναλυτική σκέψη, συνθετική ικανότητα, ικανότητες επικοινωνίας και συνεργασίας). Χρησιμοποιούνται περιβάλλοντα οπτικού προγραμματισμού όπως είναι το Scratch και το Kodu.

Στο Γυμνάσιο (ΥΠΔΒΜΘ, 2011β) η ενότητα του προγραμματισμού προσεγγίζεται μέσα από τη γεωμετρία της χελώνας είτε με τη γλώσσα Logo είτε με οπτικά περιβάλλοντα όπως το Scratch. Βασικός στόχος της ενότητας αυτής είναι η σταδιακή εξοικείωση των μαθητών με τον προγραμματισμό μέσα από την αξιοποίηση διαθέσιμων εκπαιδευτικών περιβαλλόντων οπτικού προγραμματισμού. Αξιοποιείται επίσης η διαδραστική προσομοίωση σχεδίασης απλών γεωμετρικών σχημάτων με Logo. Επίσης στην τελευταία τάξη γίνεται μια εισαγωγή στην υπολογιστική σκέψη, χωρίς να αναφέρεται ο όρος αυτός, μέσα από αλγοριθμικά προβλήματα όπως η έξοδος από λαβύρινθο και οι πύργοι του Ανόι.

## *2.4 Υπολογιστική Σκέψη*

Όπως διαφαίνεται λοιπόν υπάρχει μια διεθνής τάση εισαγωγής της υπολογιστικής σκέψης μέσα από το μάθημα της πληροφορικής και μια μετατόπιση του μαθήματος σε πιο προχωρημένα θέματα προγραμματισμού και σχεδίασης αλγορίθμων, στην δευτεροβάθμια εκπαίδευση. Η Υπολογιστική Σκέψη (ΥΣ) αναγνωρίζεται ως μια

βασική ικανότητα η οποία συμπληρώνει τις τρεις άλλες που είναι η ανάγνωση, η γραφή και τα μαθηματικά (Wing, 2006) και περιλαμβάνει όλες εκείνες τις δεξιότητες που απαιτούνται για την επίλυση προβλήματος όπως η ανάλυση σε υποπροβλήματα, η γενίκευση, η αφαίρεση και η σύνθεση των λύσεων των υποπροβλημάτων. Ο αρχικός ορισμός της Wing (2006) για την υπολογιστική σκέψη έχει ως εξής:

*Η υπολογιστική σκέψη αφορά στην επίλυση προβλημάτων, το σχεδιασμό συστημάτων και την κατανόηση της ανθρώπινης συμπεριφοράς, χρησιμοποιώντας έννοιες που είναι θεμελιώδους σημασίας για την επιστήμη των υπολογιστών.*

Ένας πιο πρόσφατος ορισμός από την ένωση καθηγητών πληροφορικής (CSTA) της ACM αναφέρει:

*Η υπολογιστική σκέψη είναι μια μεθοδολογία επίλυσης προβλημάτων η οποία επεκτείνει την επιρροή της Επιστήμης των Υπολογιστών σε όλα τα πεδία διότι παρέχει μέσα ανάλυσης και ανάπτυξης λύσεων σε προβλήματα τα οποία μπορούν να λυθούν υπολογιστικά.*

Η υπολογιστική σκέψη περιλαμβάνει επίσης την αναπαράσταση, οργάνωση και ανάλυση των δεδομένων. Η σωστή μοντελοποίηση του προβλήματος και η κωδικοποίηση των δεδομένων μπορεί να καθορίσουν το είδος του αλγορίθμου επίλυσης του προβλήματος. Δεν αρκεί απλά να λύσουμε ένα πρόβλημα αλλά να βρούμε την βέλτιστη λύση του.

Έχουν δοθεί διάφοροι ορισμοί για την υπολογιστική σκέψη. Αυτό που μένει αναλλοίωτο σε όλους τους ορισμούς είναι η ικανότητα σχεδίασης αλγορίθμων για την επίλυση προβλημάτων. Η έμφαση πλέον στα σύγχρονα προγράμματα σπουδών δεν δίνεται στην εκμάθηση κάποιας γλώσσας προγραμματισμού αλλά στη σχεδίαση αλγορίθμων, σε δεξιότητες γενίκευσης και αναπαράστασης πληροφορίας (Lu & Fletcher, 2009). Οι μαθητές έχουν ήδη κάποιες δεξιότητες αναλυτικής σκέψης και επίλυσης προβλήματος όπως έδειξαν οι Simon et al. (2006). Το ίδιο συμπέρασμα προέκυψε και από δυο έρευνες που έγιναν στο πλαίσιο της διατριβής (Βραχνός & Τζιμογιάννης, 2016· 2018). Αυτές οι δεξιότητες υπολογιστικής σκέψης μπορούν να χρησιμοποιηθούν σε διαφορετικά γνωστικά αντικείμενα όπως φυσικές επιστήμες, οικονομικές επιστήμες, μαθηματικά και άλλα.

Στην επιστήμη της πληροφορικής η ικανότητα ενός μαθητής να τοποθετήσει σε αύξουσα σειρά με βάση κάποιο χαρακτηριστικό μια ακολουθία αντικειμένων αποτελεί ένα καλό παράδειγμα υπολογιστική σκέψης, αφού ο μαθητής έχει σχεδιάσει στο μυαλό

του έναν αλγόριθμο ταξινόμησης. Το ίδιο ισχύει όταν κάποιος ψάχνει ένα όνομα στον τηλεφωνικό κατάλογο, αξιοποιώντας την αλφαβητική σειρά ή το ευρετήριο. Ουσιαστικά σχεδιάζει στο μυαλό του έναν αλγόριθμο αναζήτησης σε μια δική του ψευδογλώσσα.

## *2.5 Ο ρόλος των αναπαραστάσεων και εποικοδομισμός των προγραμματιστικών εννοιών*

Οι αναπαραστάσεις που σχηματίζουν οι μαθητές για τα προγραμματιστικά αντικείμενα οικοδομούνται πάνω σε προϋπάρχουσες γνώσεις που ήδη έχουν είτε από το σχολείο είτε από την καθημερινότητά τους, τις προ-αναπαραστάσεις (Κόμης 2005), οι οποίες συνήθως δεν αποτελούν προϊόν συστηματικής διδακτικής πράξης. Κάποιες από αυτές τις αναπαραστάσεις μπορούν να βοηθήσουν στην οικοδόμηση νέων εννοιών, όμως αρκετές συνιστούν σημαντικά γνωστικά και επιστημολογικά εμπόδια στην μαθησιακή διαδικασία. Ειδικά οι λανθασμένες προ-αναπαραστάσεις μπορούν να οδηγήσουν σε σημαντικές παρανοήσεις τους μαθητές. Για παράδειγμα πολλοί μαθητές πιστεύουν ότι η προγραμματιστική μεταβλητή διατηρεί όλες τις τιμές που έχει λάβει και όχι μόνο την τελευταία, μια παρανόηση γνωστή ως παρανόηση στοίβας (Jimoyiannis 2011· Vrachnos & Jimoyiannis 2017).

Το σύγχρονο παιδαγωγικό και μαθησιακό πλαίσιο βασίζεται την θεωρία του εποικοδομισμού (constructivism). Σύμφωνα με την θεωρία αυτή η μάθηση δεν είναι αποτέλεσμα μετάδοσης της γνώσης, αλλά μια ενεργητική διαδικασία οικοδόμησης, αναδόμησης ή επέκτασης της γνώσης η οποία βασίζεται στις εμπειρίες που έχουν αποκτήσει οι μαθητές από τον πραγματικό κόσμο. Δηλαδή πολλές νέες έννοιες οικοδομούνται σε προϋπάρχουσες έννοιες ή εμπειρίες, αφού οι μαθητές πριν ακόμα διδαχθούν κάποιο αντικείμενο διαθέτουν ήδη κάποιες γνώσεις ή εμπειρίες σχετικά με αυτό. Ο σχεδιασμός των διδακτικών στρατηγικών που θα πρέπει να ακολουθηθούν θα πρέπει να λαμβάνει υπόψη τις προϋπάρχουσες γνώσεις των μαθητών στο γνωστικό αντικείμενο της διδασκαλίας. Για να γίνει όμως αυτό θα πρέπει πρώτα να εντοπιστούν οι γνώσεις αυτές.

Η οικοδόμηση της γνώσης πραγματοποιείται μέσα από διαδικασίες διερεύνησης της πραγματικότητας και συλλογικές διεργασίες με υψηλό βαθμό αλληλεπίδρασης μεταξύ των συμμετεχόντων. Μέσα από την αλληλεπίδραση εμφανίζονται γνωστικές συγκρούσεις οι οποίες συμβάλλουν στην οικοδόμηση της γνώσης.

Ο Seymour Papert (1972) επέκτεινε τη θεωρία του εποικοδομισμού στον *κατασκευαστικό εποικοδομισμό* (constructionism), όπου προσθέτει ότι η οικοδόμηση νέας γνώσης συντελείται πιο αποτελεσματικά μέσα από δραστηριότητες στις οποίες ο μαθητής κατασκευάζει κάτι απτό. Αυτό θα μπορούσε να είναι μια κατασκευή ή ακόμα και ένα πρόγραμμα στον υπολογιστή. Σύμφωνα με τη θεωρία αυτή οι μαθητές οικοδομούν νέα γνώση όταν ενεργά ασχολούνται με την κατασκευή κάποιου τεχνήματος (artifact) ή αντικειμένου. Στην περίπτωση του προγραμματισμού, το κατασκεύασμα είναι ένα πρόγραμμα υπολογιστή και το εργαλείο είναι η γλώσσα προγραμματισμού.

Ο σχεδιασμός προγραμμάτων μπορεί να γίνει μέσα από κατάλληλα σχεδιασμένες δραστηριότητες διερευνητικού χαρακτήρα χωρίς όμως να περιορίζεται η ελευθερία στη δημιουργικότητα του μαθητή. Για αυτό ο βαθμός καθοδήγησης σε αυτές τις δραστηριότητες παίζει σημαντικό ρόλο. Η αλληλεπίδραση του μαθητή με λογισμικά που υποστηρίζουν διερευνητικές διαδικασίες μπορεί να μειώσει σε μεγάλο βαθμό την καθοδήγηση από τον εκπαιδευτικό και βαθμιαία να οδηγήσει τον μαθητή στην οικοδόμηση της νέας γνώσης μέσα από μια προσωπική διαδρομή.

## *2.6 Διδακτική του Προγραμματισμού*

Η σημασία του προγραμματισμού ως γνωστική δραστηριότητα, η συνεισφορά του στην ανάπτυξη δομημένης σκέψης και η επίδρασή του στην καλλιέργεια πνευματικών δεξιοτήτων υψηλού επιπέδου είχε τεθεί για πρώτη φορά από τον Papert (1980) και έχει επιβεβαιωθεί από διάφορες έρευνες (Ennis, 1994· Charlton & Birkett, 1994· Kagan, 1989). Για μια μεγάλη περίοδο η διδασκαλία του προγραμματισμού είχε ταυτιστεί με τη διδασκαλία μιας συγκεκριμένης γλώσσας προγραμματισμού, μια αντίληψη που σήμερα έχει ξεπεραστεί. Από την δεκαετία του 1990, οι ερευνητές ασχολούνται με τη διδασκαλία βασικών προγραμματιστικών εννοιών όπως είναι η μεταβλητή, οι δομές ελέγχου, οι δομές δεδομένων κ.α., ανεξάρτητα από τη γλώσσα που έχει επιλεγεί για την παρουσίαση αυτών των εννοιών. Μια διαφοροποίηση υπάρχει μόνο όταν αλλάζουμε προγραμματιστικό παράδειγμα π.χ. η μετάβαση από τον δομημένο προγραμματισμό στον αντικειμενοστρεφή ή ακόμα και στον συναρτησιακό δεν είναι κάτι απλό.

Η διδασκαλία του προγραμματισμού στοχεύει στην πρόσκτηση και την αποτελεσματική εφαρμογή τριών αλληλεξαρτώμενων μορφών γνώσης (Bayman & Mayer 1988· Κόμης, 2005· Τζιμογιάννης 2005):

- **Συντακτική γνώση**: Είναι η γνώση των ειδικών χαρακτηριστικών μιας γλώσσας προγραμματισμού και των κανόνων χρήσης (γραμματική, συντακτικό, αλφάβητο).
- **Εννοιολογική γνώση**: Αφορά την κατανόηση των προγραμματιστικών δομών και αντικειμένων και διακρίνεται στην **σημασιολογική** (semantic) και τη **σχηματική** (schematic) γνώση. Η σημασιολογική γνώση εξετάζει εννοιολογικά τις προγραμματιστικές έννοιες της μεταβλητής, της δομής επανάληψης των δομών δεδομένων κλπ. Η σχηματική γνώση συνίσταται στο σύνολο των εντολών (instruction set), είτε αυτές είναι εντολές της γλώσσας (statements) είτε συναρτήσεις και διαδικασίες βιβλιοθηκών της γλώσσας (APIS) που υλοποιούν χρήσιμους αλγορίθμους. Σε αντίθεση με τους αρχάριους, οι έμπειροι προγραμματιστές έχουν ένα καλά οργανωμένο σύνολο εντολών τις οποίες μπορούν εύκολα να ανακαλέσουν και να χρησιμοποιήσουν για την επίλυση ενός προβλήματος.
- **Στρατηγική γνώση (μεταγνώση):** Αναφέρεται στην ικανότητα εφαρμογής των παραπάνω γνώσεων για την επίλυση αυθεντικών προβλημάτων στον προγραμματισμό. Βασίζεται στην ανάπτυξη δεξιοτήτων υψηλού επιπέδου για τον σχεδιασμό αλγορίθμων, την υλοποίηση προγραμμάτων και την ικανότητα μεταφοράς δεξιοτήτων επίλυσης προβλημάτων μεταξύ διαφορετικών γνωστικών αντικειμένων.

Η διδασκαλία του προγραμματισμού δεν έχει να κάνει με τη διδασκαλία του συντακτικού και της σημασιολογίας μιας γλώσσας αλλά με τη χρήση των δομικών εργαλείων που παρέχει η γλώσσα για την επίλυση ενός προβλήματος. Αυτό δεν είναι εύκολο γιατί η ανάπτυξη ενός προγράμματος απαιτεί τον χειρισμό πολλών αφηρημένων οντοτήτων όπως είναι οι λογικές τιμές, η αναδρομή, οι δομές δεδομένων κλπ. Οι αρχάριοι προγραμματιστές αντιμετωπίζουν πολλές δυσκολίες ξεκινώντας από την έννοια της μεταβλητής πάνω στην οποία χτίζονται αφηρημένες προγραμματιστικές έννοιες όπως είναι η έννοια του μετρητή της επανάληψης (loop counter), του δείκτη πίνακα (array index), του αθροιστή/συσσωρευτή, του δείκτη στη μνήμη (memory pointer). Όλες οι παραπάνω αφηρημένες έννοιες όχι μόνο δεν μπορούν να περιγραφούν με όρους από την καθημερινή εμπειρία του μαθητή, αλλά είναι πολύ δύσκολο να αναπαρασταθούν με παραδοσιακά διδακτικά μέσα.

Οι αρχάριοι προγραμματιστές (μαθητές ή φοιτητές) έχουν συνήθως ελάχιστες γνώσεις προγραμματισμού που είναι επιφανειακές και όχι επαρκώς οργανωμένες (Bonar & Soloway, 1985). Επίσης παρουσιάζουν δυσκολίες στην οικοδόμηση επιστημονικά ορθών νοητικών μοντέλων για τις προγραμματιστικές δομές. Ο Winslow (1996) θεωρεί ότι η χρονική περίοδος που απαιτείται για να γίνει ένας αρχάριος προγραμματιστής έμπειρος είναι περίπου δέκα έτη. Από την άλλη οι έμπειροι προγραμματιστές χρησιμοποιούν υψηλότερα επίπεδα αφαίρεσης, και ικανότητες που έχουν κατακτηθεί μέσα από χρόνια εμπειρίας στην επίλυση προγραμματιστικών προβλημάτων. Από τα παραπάνω συνάγεται ότι οι αρχάριοι προγραμματιστές χρειάζονται υποστήριξη είτε με τη μορφή στοχευμένων διδακτικών παρεμβάσεων είτε με τη μορφή εκπαιδευτικών περιβαλλόντων προγραμματισμού.

## *2.7 Δυσκολίες – παρανοήσεις των μαθητών στον προγραμματισμό*

### *2.7.1 Εισαγωγή*

Παρόλο που ο προγραμματισμός αποτελεί μια δραστηριότητα η οποία είναι εξ' ορισμού βασισμένη στον εποικοδομισμό και τη διερευνητική μάθηση, θεωρείται αρκετά δύσκολη δραστηριότητα για τους αρχάριους προγραμματιστές. Τα μαθήματα προγραμματισμού που προσφέρονται στις σχολές πληροφορικής θεωρούνται πολύ απαιτητικά και αρκετοί φοιτητές δεν καταφέρνουν να τα ολοκληρώσουν επιτυχώς (Bennedsen & Caspersen, 2007). Μια έρευνα που έγινε σε τέσσερα πανεπιστήμια (McCracken et al., 2001) έδειξε ότι πολλοί φοιτητές δεν μπορούν να σχεδιάσουν ένα πρόγραμμα για την επίλυση ενός προβλήματος, μετά την ολοκλήρωση ενός εισαγωγικού μαθήματος προγραμματισμού. Μια άλλη έρευνα που έγινε σε επτά χώρες (Lister et al., 2004) κατέληξε σε παρόμοια αποτελέσματα και έδειξε ότι πολλοί φοιτητές δυσκολεύονται ακόμα και να εκτελέσουν με χαρτί και μολύβι ένα πρόγραμμα.

Η επισκόπηση της βιβλιογραφίας δείχνει ότι οι αρχάριοι προγραμματιστές αντιμετωπίζουν σοβαρά προβλήματα στην κατανόηση βασικών προγραμματιστικών εννοιών (Hoc et al., 1990· Samurçay, 1989· Soloway & Spohrer, 1989). Επίσης είναι κοινά παραδεκτό ότι σχηματίζουν λανθασμένα ή ημιτελή νοητικά μοντέλα σχετικά με προγραμματιστικές δομές και αντικείμενα ενώ δυσκολεύονται να κατανοήσουν τα βασικά μέρη ενός προγράμματος και τη σχέση μεταξύ τους, πόσο μάλλον να σχεδιάσουν έναν νέο αλγόριθμο για την επίλυση ενός προβλήματος (Robins, Rountree, & Rountree, 2003). Γενικότερα οι έμπειροι προγραμματιστές μπορούν να διακρίνουν

τη λειτουργία ενός προγράμματος σε ένα πιο αφηρημένο επίπεδο ενώ οι αρχάριοι προσεγγίζουν το πρόγραμμα εντολή-εντολή χωρίς να μπορούν να διακρίνουν το γενικότερο σκοπό του.

Τα νοητικά μοντέλα που σχηματίζουν και ο τρόπος με τον οποίο αντιλαμβάνονται τις αφηρημένες προγραμματιστικές δομές, όπως είναι οι μεταβλητές, οι δομές ελέγχου και τα αντικείμενα αποτελούν μερικά από τα βασικά θέματα της έρευνας στη διδακτική του προγραμματισμού (Corney et al., 2011· Ma et al., 2011· Sheard et al., 2008). Ενώ οι έμπειροι προγραμματιστές αναπτύσσουν νοητικούς μηχανισμούς για την αναπαράσταση αυτών των αντικειμένων, οι αρχάριοι προγραμματιστές επιδεικνύουν μια επιφανειακή κατανόηση των εννοιών αυτών κατά την επίλυση προβλημάτων. Αυτό μπορεί να οδηγήσει σε ελλιπή εννοιολογικά μοντέλα τα οποία με τη σειρά τους ευθύνονται για ενδεχόμενες παρανοήσεις πάνω στις δομές αυτές (Ma et al., 2011). Ο εντοπισμός και η ερμηνεία των παρανοήσεων θα βοηθήσει τους καθηγητές ώστε να προσαρμόσουν ανάλογα τη διδασκαλία τους και τους μαθητές να σχηματίσουν ορθά και πλήρη εννοιολογικά μοντέλα για αυτές τις προγραμματιστικές έννοιες.

Οι Soloway & Spohrer (1989) παρουσιάζουν διάφορες παρανοήσεις των αρχάριων προγραμματιστών πάνω σε θεμελιώδεις προγραμματιστικές δομές όπως είναι οι μεταβλητές και οι εντολές επανάληψης. Πιο πρόσφατες έρευνες μελέτησαν τις παρανοήσεις των μαθητών στις μεταβλητές και τις εντολές ανάθεσης τιμής σε αυτές (Jimoyiannis 2011· Ma et al. 2011· Sajaniemi & Kuittinen, 2005) ενώ άλλοι (Fleury, 1991· Madison & Gifford 2003), εστίασαν στις παραμέτρους, στις δομές επιλογής και επανάληψης και την αναδρομή (Putnam et al., 1989· Soloway & Spohrer, 1989). Οι Sirkiä & Sorva (2012) κατέγραψαν διάφορες παρανοήσεις που έχουν οι αρχάριοι προγραμματιστές μέσα από ασκήσεις οπτικοποιήσεων.

Ανιχνεύθηκαν οι περισσότερες από τις παρανοήσεις που έχουν αναφερθεί παραπάνω, όπως:

- η παρανόηση ιστορικού όπου μια μεταβλητή μπορεί να αποθηκεύει ταυτόχρονα περισσότερες από μια τιμές
- στην εντολή απόδοσης τιμής η τιμή μεταφέρεται από τη μια μεταβλητή στην άλλη.

Ωστόσο οι περισσότερες εργασίες για παρανοήσεις των μαθητών στον προγραμματισμό τα τελευταία χρόνια εστιάζονται σε έννοιες του αντικειμενοστρεφούς προγραμματισμού (Boustedt, 2012· Fleury, 2000· Holland, Griffiths & Goodman,

1997· Hristova et. al., 2003), κυρίως λόγω της διάδοσης που έχουν οι αντικειμενοστρεφείς γλώσσες (Java, C++), σε εισαγωγικά μαθήματα προγραμματισμού. Έχει δημοσιευθεί πληθώρα εργασιών για τις δυσκολίες που αντιμετωπίζουν οι μαθητές με θεμελιώδεις αντικειμενοστρεφείς έννοιες όπως είναι οι κλάσεις, οι μέθοδοι, οι κατασκευαστές, οι αναφορές και άλλες (Eckerdal & Thune, 2005· Ragonis & Ben-Ari, 2005· Thomasson, Ratcliffe, & Thomas, 2006).

### *2.7.2 Η νοητή μηχανή*

Πριν μελετήσουμε τις παρανοήσεις που έχουν οι μαθητές για διάφορες προγραμματιστικές έννοιες θα πρέπει να εξετάσουμε πως αντιλαμβάνονται το μοντέλο εκτέλεσης ενός αλγορίθμου σε κάποιον υπολογιστή. Η νοητή μηχανή είναι ένα θεωρητικό υπολογιστικό μοντέλο το οποίο ορίζεται από τη γλώσσα προγραμματισμού που χρησιμοποιούμε. Ο du Boulay(1986) που ανέφερε πρώτος τον όρο "νοητή μηχανή" τον περιέγραψε ως εξής:

*Ένα αφαιρετικό μοντέλο του υπολογιστή το οποίο μπορεί να χρησιμοποιήσει κάποιος ώστε να σκέφτεται τι μπορεί να κάνει ένας υπολογιστής. Η νοητή μηχανή χαρακτηρίζεται από τις γενικές ιδιότητες της μηχανής που πρέπει να ξέρει κάποιος ώστε να μπορεί να την ελέγχει.*

Δηλαδή η νοητή μηχανή αποτελεί ένα αφαιρετικό μοντέλο της λειτουργικότητας της μηχανής που προγραμματίζουμε, η οποία περιλαμβάνει το σύνολο όλων των επιτρεπτών εντολών. Θα μπορούσε να οριστεί ως μια νοητή διεπαφή (notional interface) που περιγράφει τη λειτουργικότητα της προγραμματιζόμενης μηχανής.

Συνοψίζοντας, μια νοητή μηχανή :

- Είναι μια ιδεατή αφαίρεση του περιβάλλοντος εκτέλεσης των προγραμμάτων και δεν αναφέρεται πάντα στο υλικό του υπολογιστή.
- Στοχεύει στην κατανόηση των διεργασιών που συντελούνται κατά την εκτέλεση ενός αλγόριθμου.
- Συνδέεται με κάποια συγκεκριμένη γλώσσα προγραμματισμού η περιβάλλον εκτέλεσης.
- Επιτρέπει την περιγραφή των σημασιολογικών χαρακτηριστικών ενός προγράμματος.

Το μοντέλο της νοητής μηχανής έχει χρησιμοποιηθεί σε πολλές έρευνες για την διευκόλυνση της ερμηνείας των παρανοήσεων και των δυσκολιών των αρχάριων

προγραμματιστών. Μια εκτενής αναφορά γίνεται στην εργασία των Robins, Rountree & Rountree (2003). Για παράδειγμα ο Mayer (1989) έδειξε ότι οι φοιτητές που είχαν σχηματίσει ένα μοντέλο νοητής μηχανής είχαν καλύτερα αποτελέσματα στην επίλυση προβλημάτων από τους φοιτητές που δεν χρησιμοποίησαν ένα τέτοιο μοντέλο.

Σύμφωνα με τον du Boulay οι βασικές προϋποθέσεις για να διευκολυνθεί η μαθησιακή διαδικασία από τη χρήση μιας νοητής μηχανή είναι οι εξής:

- η νοητή μηχανή πρέπει να αποτελεί ένα απλοποιημένο μοντέλο λειτουργίας του υπολογιστή και
- να υποστηρίζεται από κάποιο εργαλείο που θα επιτρέπει την παρουσίαση της λειτουργίας της με κάποια γραφική αναπαράσταση. Τέτοια εργαλεία αποτελούν τα λογισμικά οπτικοποίησης λογισμικού (software visualization).

Εδώ θα πρέπει να συμπληρώσουμε και μια τρίτη προϋπόθεση:

- Η απλοποίηση-αφαίρεση θα πρέπει να γίνεται με εξαιρετική προσοχή έτσι ώστε να μην προκαλεί παρανοήσεις στους μαθητές, λόγω ασυνέπειας την νοητής με την πραγματική μηχανή.

Παράλληλα με τις προσπάθειες πολλών ερευνητών (Berry & Kölling, 2016) να ορίσουν απλοποιημένα μοντέλα νοητών μηχανών για την καλύτερη παρουσίαση των προγραμματιστικών εννοιών, αναπτύσσονται διάφορα συστήματα προσομοίωσης της νοητής μηχανής μέσω διαδραστικών οπτικοποιήσεων, όπως είναι το Jeliot (Ben-Bassat Levy & Ben-Ari, 2009) και το BlueJ (Berry & Kölling, 2013) για προγράμματα στη γλώσσα Java, το UUhistle (Sorva & Sirkiä, 2010) και το Python Tutor (Guo, 2012) για προγράμματα στη γλώσσα Python και άλλα.

### *2.7.3 Η έννοια της μεταβλητής: δυσκολίες και παρανοήσεις*

Η πρώτη έννοια με την οποία έρχονται σε επαφή οι μαθητές σε ένα εισαγωγικό μάθημα προγραμματισμού είναι η έννοια της μεταβλητής. Μόνη εξαίρεση σε αυτό το γεγονός αποτελούν οι συναρτησιακές γλώσσες οι οποίες όμως χρησιμοποιούνται πολύ σπάνια σε εισαγωγικά μαθήματα προγραμματισμού. Η μεταβλητή αποτελεί μια θεμελιώδης έννοια πάνω στην οποία οικοδομούνται άλλες πιο σύνθετες έννοιες όπως είναι αυτές των μετρητών και των βρόχων, των πινάκων και άλλων δομών δεδομένων. Για αυτόν τον λόγο η μελέτη των αναπαραστάσεων των μαθητών για τις μεταβλητές αποκτά κρίσιμη σημασία στη διδακτική του προγραμματισμού και της αλγοριθμικής.

Πολλές έρευνες έχουν δείξει ότι οι μαθητές της δευτεροβάθμιας εκπαίδευσης έχουν πολλές δυσκολίες στον χειρισμό μεταβλητών κατά την επίλυση προβλημάτων (du Boulay, 1986· Jimoyiannis, 2011· Ma et al., 2011· Sajaniemi, 2002· Soloway & Spohrer, 1989). Οι μαθητές συναντούν αρκετές δυσκολίες στην κατανόηση της έννοιας της μεταβλητής, ενώ οι παρανοήσεις παραμένουν ακόμα και μετά από αρκετά μαθήματα.

Η μεταβλητή αντιστοιχεί σε μια θέση μνήμης του υπολογιστή στην οποία η προσπέλαση γίνεται με ένα όνομα που επιλέγει ο προγραμματιστής. Μια μεταβλητή μπορεί να αλλάξει περιεχόμενο πολλές φορές κατά την εκτέλεση του προγράμματος. Η κοινή αντίληψη των μαθητών για τη μεταβλητή (που συνήθως οφείλεται σε ανεπαρκείς διδακτικές προσεγγίσεις) βασίζεται στην αναλογία του κουτιού ή του γραμματοκιβωτίου, η οποία όμως είναι αιτία διάφορων παρανοήσεων για τους αρχάριους προγραμματιστές (Jimoyiannis, 2011).

Μια από αυτές είναι ότι μια μεταβλητή διατηρεί/θυμάται όλες τις τιμές τις οποίες έχει πάρει κατά τη διάρκεια του προγράμματος. Η αναπαράσταση της μεταβλητής σε αυτή την περίπτωση μοιάζει με μια εικόνα τύπου λίστας, σωρού ή στοίβας απ' όπου μπορούν να ανακτηθούν όλες οι προηγούμενες τιμές. Ο Jimoyiannis (2011) χρησιμοποιεί τον όρο "*παρανόηση στοίβας τιμών*" (stack variable misconception) (Jimoyiannis, 2011) όταν θέλει να αναφερθεί στη συγκεκριμένη παρανόηση. Στην διατριβή αυτή θα αναφερόμαστε σε αυτή την παρανόηση και ως παρανόηση "*ιστορικού*", αφού αφορά στη διατήρηση του ιστορικού όλων των τιμών που έχει λάβει η μεταβλητή.

Αυτή η παρανόηση εμφανίζεται συνήθως όταν χρησιμοποιούμε μια μεταβλητή για να διαβάζουμε ροές δεδομένων (data streams), δηλαδή έχουμε συνεχή ροή δεδομένων τα οποία δεν θέλουμε ή δεν μπορούμε να αποθηκεύσουμε στη μνήμη αλλά τα επεξεργαζόμαστε χωριστά καθώς εισάγονται. Ένα τέτοιο παράδειγμα φαίνεται στο Τμήμα Κώδικα 2.1.

Ενώ στις μεταβλητές *βαθμός* και *όνομα* έχουν καταχωρηθεί στο τέλος μόνο τα στοιχεία του 30$^{ου}$ μαθητή, πολλοί μαθητές θεωρούν ότι κάθε μια από τις δύο μεταβλητές έχει με κάποιον τρόπο κρατήσει όλο το ιστορικό των τιμών που έχει λάβει και τις οποίες μπορεί να προσπελάσει όποτε θέλει. Δηλαδή η μεταβλητή συμπεριφέρεται ως λίστα ή στοίβα.

```
Για μαθητή από 1 μέχρι 30        # Για 30 μαθητές
    Διάβασε βαθμός, όνομα        # εισάγονται τα στοιχεία ενός μαθητή κάθε φορά
Τέλος_Επανάληψης

Για μαθητή από 1 μέχρι 30
    Αν βαθμός > 18 Τότε
        Γράψε "Ο ", όνομα, " πήρε Άριστα "
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 2.1 . Παρανόηση ιστορικού τιμών (stack variable misconception)**

Επίσης έχει παρατηρηθεί (Jimoyiannis, 2011· Soloway & Spohrer, 1989) ότι πολλοί μαθητές παρουσιάζουν δυσκολίες κατά τον χειρισμό συγκεκριμένων τύπων μεταβλητών. Ενώ οι αριθμητικές μεταβλητές χειρίζονται από τους μαθητές σχετικά εύκολα, αφού παραπέμπουν σε οικεία γνωστικά σχήματα, η χρήση αλφαριθμητικών, λογικών ή πιο σύνθετων δομών δεδομένων απαιτεί την οικοδόμηση νέων αναπαραστάσεων για τις οποίες οι μαθητές συναντούν σοβαρές δυσκολίες.

Οι αναπαραστάσεις που έχουν οι μαθητές/φοιτητές σχετικά με την έννοια της μεταβλητής φαίνονται από τον τρόπο με τον οποίο χρησιμοποιούν την εντολή ανάθεσης τιμής (εκχώρησης) μεταξύ μεταβλητών. Αρκετές έρευνες (Corney et al., 2011· Jimoyiannis, 2011) διαπιστώνουν σοβαρές δυσκολίες των μαθητών όταν και στα δύο πλευρές της εντολής εκχώρησης είναι μεταβλητή. Πολλοί μαθητές έχουν την παρανόηση ότι η τιμή της μεταβλητής στο δεξί μέλος της εντολής δεν αντιγράφεται αλλά μεταφέρεται στην μεταβλητή στο αριστερό μέλος της εκχώρησης. Για παράδειγμα η μεταβλητή Α δεν θα έχει τιμή μετά την παρακάτω εντολή εκχώρησης ενώ η Β θα είναι ίση με 496.

```
B ← 496

A ← B
```

Για αυτή την παρανόηση ευθύνεται το μοντέλο του κουτιού/γραμματοκιβωτίου που έχει επικρατήσει για τη διδασκαλία της έννοιας της μεταβλητής. Το μοντέλο αυτό δεν αποτελεί ακριβή αναπαράσταση της έννοιας της μεταβλητής, αφού αν πάρουμε το περιεχόμενο ενός κουτιού και το τοποθετήσουμε σε ένα άλλο, το περιεχόμενο δεν αντιγράφεται αλλά μεταφέρεται. Το μοντέλο του γραμματοκιβωτίου μπορεί να

προκαλέσει και άλλα προβλήματα αργότερα στην κατανόηση της έννοιας της αναφοράς που συναντάται στη γλώσσα Java, η οποία χρησιμοποιείται ευρέως σε εισαγωγικά μαθήματα προγραμματισμού. Στην περίπτωση αυτή είναι πιθανό πολλές μεταβλητές να περιέχουν αναφορές στην ίδια θέση μνήμης, κάτι που μπορεί να δυσκολέψει τους μαθητές αφού δεν είναι δυνατόν πολλά γραμματοκιβώτια να περιέχουν το ίδιο ακριβώς γράμμα.

Άλλοι θεωρούν ότι με κάποιο τρόπο οι μεταβλητές είναι συνδεδεμένες συγχέοντας την έννοια της μαθηματικής μεταβλητής που έχουν ήδη διδαχθεί. Επίσης συγχέουν τον τελεστή της ισότητας από τα μαθηματικά με τον τελεστή ανάθεσης τιμής στον προγραμματισμό. Αυτό συμβαίνει γιατί κάποιες γλώσσες χρησιμοποιούν τον τελεστή "=" για την ανάθεση τιμής σε μεταβλητή, π.χ. C, C++, Java, Python, σε αντίθεση με την ψευδογλώσσα που πολλές φορές χρησιμοποιεί το βέλος '←', για να δείξει ότι η έκφραση δεξιά αντιγράφεται στη μεταβλητή του αριστερού μέλους.

Μια εξήγηση για αυτό είναι ότι οι αρχάριοι προγραμματιστές μεταφέρουν τις αναπαραστάσεις και τα μοντέλα που ήδη έχουν στο πεδίο του προγραμματισμού, με αποτέλεσμα να δίνουν στην προγραμματιστική μεταβλητή ιδιότητες της μαθηματικής μεταβλητής (Jimoyiannis, 2011· Samurcay, 1989).

### *2.7.4 Οι ρόλοι των μεταβλητών*

Στο πλαίσιο της οικοδόμησης αποτελεσματικών αναπαραστάσεων για την έννοια της μεταβλητής έχει προταθεί η διδασκαλία των ρόλων μεταβλητών ως μια εναλλακτική διδακτική προσέγγιση (Sajaniemi, 2005· Sajaniemi & Kuittinen, 2005). Οι ρόλοι των μεταβλητών αφορούν σε συγκεκριμένες χρήσεις που έχουν οι μεταβλητές σε ένα πρόγραμμα. Ως ιδέα, εισήχθησαν από τον Sajaniemi (2002) ο οποίος ανέλυσε προγράμματα γραμμένα από αρχάριους προγραμματιστές. Σύμφωνα με την ανάλυση του Sajaniemi αρκούν δέκα (10) ρόλοι για να περιγράψουν το 99% της χρήσης όλων των μεταβλητών στα προγράμματα των αρχάριων προγραμματιστών. Σκοπός αυτής της προσέγγισης είναι η κατανόηση των λειτουργικών χαρακτηριστικών κάθε μεταβλητής τα οποία οριοθετούν το ρόλο της μεταβλητής στο πρόγραμμα.

Σύμφωνα με τον Sajaniemi (2005) το σύνολο αυτό περιλάμβανε τους παρακάτω ρόλους στους οποίους αργότερα προστέθηκαν και άλλοι, για να καλύψουν τις παραμέτρους στην κλήση υποπρογραμμάτων αλλά και τον αντικειμενοστρεφή προγραμματισμό:

- **Σταθερά** *(Fixed Value)*: Η μεταβλητή λαμβάνει μια αρχική τιμή που δεν αλλάζει κατά την εκτέλεση του προγράμματος.
- **Μετρητής** *(Stepper)*: Η μεταβλητή λαμβάνει μια σειρά από συγκεκριμένες τιμές οι οποίες προκύπτουν με αυτοματοποιημένο τρόπο και ακολουθούν συγκεκριμένο μαθηματικό μοτίβο.
- **Εξαρτώμενη** *(Follower)*: Η μεταβλητή, λαμβάνει νέα τιμή η οποία εξαρτάται από άλλη μεταβλητή.
- **Τρέχουσα τιμή** *(Most-recent holder)*: Η μεταβλητή περιέχει την τελευταία από μια σειρά τιμών (π.χ. η τελευταία τιμή από μια σειρά διαδοχικών τιμών εισόδου).
- **Τελική τιμή** *(Most-wanted holder)*: Η μεταβλητή περιέχει τη βέλτιστη τιμή από μια σειρά τιμών, χωρίς περιορισμούς στον ορισμό του βέλτιστου (π.χ. η μεγαλύτερη, κάθε φορά, τιμή από μια σειρά δεδομένων εισόδου).
- **Συσσωρευτής** *(Gatherer)*: Η μεταβλητή περιέχει το αποτέλεσμα διαφορετικών, ανεξάρτητων μεταξύ τους μεταβλητών (π.χ. σε ένα παιχνίδι με κάρτες, η μεταβλητή περιέχει το σύνολο των καρτών που έχει ένας παίκτης, όταν μπορεί να ρίξει κάποιες κάρτες κάθε φορά).
- **Συναρτησιακή** *(Transformation)*: Η μεταβλητή περιέχει το αποτέλεσμα διαφορετικών, ανεξάρτητων μεταξύ τους μεταβλητών (π.χ. σε ένα παιχνίδι
- **Βοηθητικός πίνακας** *(Organizer)*: Πρόκειται για πίνακα που χρησιμοποιείται για την αναδιάταξη των στοιχείων του (π.χ. ένας πίνακας για την ταξινόμηση ενός συνόλου τιμών). Ο πίνακας αυτός είναι διαφορετικός και από τον αρχικό πίνακα και από το αποτέλεσμα. Θα μπορούσε να θεωρηθεί ως ο πάγκος (πίνακας) εργασίας του συγκεκριμένου αλγορίθμου.
- **Σημαία/Φρουρός** *(One-way flag)*: Η μεταβλητή μπορεί να πάρει μόνο δύο τιμές. Όταν αλλάξει μια φορά η τιμή της, δεν αλλάζει ξανά κατά την εκτέλεση του προγράμματος.
- **Βοηθητική** *(Temporary)*: Η μεταβλητή περιέχει προσωρινά μια τιμή.

Η εναλλακτική αυτή προσέγγιση διδασκαλίας για τα εισαγωγικά μαθήματα προγραμματισμού βοηθά στη δημιουργία λειτουργικών αναπαραστάσεων για τις μεταβλητές ενός προγράμματος και επιπλέον βοηθά τους αρχάριους προγραμματιστές

να διαπιστώσουν τα λάθη τους και να κατανοήσουν εκτεταμένα ή περίπλοκα προγράμματα (Byckling & Sajaniemi, 2006).

Στη διατριβή αυτή οι ρόλοι των μεταβλητών χρησιμεύουν στην οπτικοποίηση των αλγορίθμων, όπου οι μεταβλητές-μετρητές (i,j) της επανάληψης έχουν διαφορετική γραφική αναπαράσταση από τα στοιχεία του πίνακα (A[j]) και τις τυχόν βοηθητικές μεταβλητές.

### *2.7.5 Η παρανόηση του από μηχανής θεού*

Μια πολύ γνωστή παρανόηση που έχει αναφερθεί από τον Pea(1986) είναι αυτή του "*από μηχανής θεού*" (superbug). Σε αυτή την περίπτωση η νοητή μηχανή μπορεί να σκέφτεται από μόνη της και να εκτελεί εντολές που δεν έχει δώσει ο προγραμματιστής έτσι ώστε το τελικό αποτέλεσμα να ανταποκρίνεται σε αυτό που είχε σκεφτεί ο προγραμματιστής αλλά δεν μπόρεσε να περιγράψει πλήρως. Θα μπορούσαμε να πούμε επίσης ότι η νοητή μηχανή και ο προγραμματιστής συνεργάζονται κατά κάποιο τρόπο.

Ένα τέτοιο παράδειγμα φαίνεται στο Τμήμα Κώδικα 2.2 το οποίο υπολογίζει πόσοι μαθητές έχουν τον μεγαλύτερο βαθμό σε μια τάξη 30 παιδιών.

```
μέγιστος ← -1                    # όλοι οι βαθμοί είναι θετικοί
Για μαθητή από 1 μέχρι 30
    Διάβασε βαθμός               # ο χρήστης εισάγει τους βαθμούς και
    Αν βαθμός > μέγιστος Τότε    # αποθηκεύει κάθε βαθμό στην ίδια μεταβλητή
        μέγιστος ← βαθμός
    Τέλος_Αν
Τέλος_Επανάληψης
πλήθος ← 0
Για μαθητή από 1 μέχρι 30
    Αν βαθμός = μέγιστος Τότε    # ελέγχω αν ο μέγιστος είναι ίσος με τον
        πλήθος ← πλήθος + 1      # τελευταίο βαθμό που διάβασα
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 2.2. Υπολογισμός συχνότητας της μέγιστης τιμής**

Πολλοί αρχάριοι προγραμματιστές θεωρούν ότι με κάποιο μαγικό τρόπο όλες οι τιμές που δόθηκαν έχουν αποθηκευτεί στη μεταβλητή *βαθμός* και στη συνέχεια πάλι κατά κάποιο μαγικό τρόπο η μεταβλητή-δείκτης *μαθητής*, συνδέεται κάπως με το βαθμό και σε κάθε βήμα η μηχανή γνωρίζει το όνομα του μαθητή που θέλουμε.

### *2.7.6 Δυσκολίες-Παρανοήσεις των μαθητών στη δομή επιλογής*

Η δομή ελέγχου ή δομή επιλογής είναι μια από τις βασικές λογικές δομές του προγραμματισμού και συναντάται σε όλες τις γλώσσες προγραμματισμού. Η δομή αυτή δίνει τη δυνατότητα εκτέλεσης ενός μπλοκ εντολών μόνο αν ισχύει μια λογική συνθήκη. Η δομή επιλογής είναι μια δομή που συναντούν οι αρχάριοι στα πρώτα μαθήματα του προγραμματισμού και τους δυσκολεύει αρκετά (Sirkiä & Sorva, 2012· Rogalski & He, 1989· Soloway & Spohrer, 1989· Veerasamy, D'Souza, & Laakso, 2016).

Στη βιβλιογραφία υπάρχουν λίγες αναφορές στις δυσκολίες των μαθητών με τη δομή επιλογής (Bayman & Mayer, 1983· du Boulay· 1986, Putnam et al., 1986). Σε μια από αυτές ο du Boulay(1986) αναφέρει ότι οι μαθητές δυσκολεύονται να κατανοήσουν ότι οι εντολές που βρίσκονται αμέσως μετά τη δομή επιλογής εκτελούνται πάντα ανεξάρτητα από την αποτίμηση της συνθήκης. Ένα παράδειγμα λανθασμένης χρήσης της δομής επιλογής φαίνεται στο Τμήμα Κώδικα 2.3.

```
1: Αν βαθμός >= 10 Τότε
        Γράψε "Ο ", όνομα, " προβιβάστηκε "
   Τέλος_Αν
2: Γράψε  "Ο ", όνομα, " θα επαναλάβει την τάξη "
```

**Τμήμα Κώδικα 2.3. Λανθασμένη χρήση της δομής επιλογής**

Συνήθως τα περισσότερα προβλήματα που παρατηρούνται στη δομή επιλογής έχουν να κάνουν με τη συνδυαστική χρήση της εντολής Goto η οποία πλέον δεν χρησιμοποιείται. Επίσης πολλοί μαθητές παρουσιάζουν δυσκολίες στην κατανόηση της σειράς εκτέλεσης των εντολών του προγράμματος όταν εμπλέκονται εμφωλευμένες (embedded) δομές επιλογής, όπως και στην σύνταξη της σωστής συνθήκης (Putnam et al., 1986). Δύο από τα βασικότερα εμπόδια είναι η χρήση λογικών τελεστών και η δημιουργία σύνθετων λογικών εκφράσεων μαζί με την αντίληψη που έχουν οι αρχάριοι προγραμματιστές την σειριακή εκτέλεση ενός προγράμματος.

Σημαντική παράμετρο στην κατανόηση της δομής ελέγχου διαδραματίζουν και οι πρότερες γνώσεις μαθηματικών και λογικής (Bonar & Soloway, 1985). Οι έρευνες δείχνουν ότι οι μαθητές που έχουν ένα ανεπτυγμένο μαθηματικό υπόβαθρο, εξοικειώνονται πιο γρήγορα με τη λειτουργία της δομής επιλογής

### *2.7.7 Δυσκολίες-Παρανοήσεις των μαθητών στις δομές επανάληψης*

Η επανάληψη (iteration) ή βρόχος (loop) επιτρέπει την επανάληψη ενός μπλοκ εντολών κάποιες φορές, ο αριθμός των οποίων μπορεί να μην είναι γνωστός εκ των προτέρων. Τρεις συνηθισμένες μορφές της δομής επανάληψης είναι οι ακόλουθες:

```
                    i ← 1                       i ← 1
Για i από 1 μέχρι 100   Όσο i <= 100 Επανάλαβε   Αρχή_Επανάληψης
  Γράψε i                 Γράψε i                   Γράψε i
Τέλος_Επανάληψης          i ← i + 1                 i ← i + 1
                        Τέλος_Επανάληψης          Μέχρις_ότου  i > 100
```

**Τμήμα Κώδικα 2.4. Οι τρεις βασικές δομές επανάληψης**

Όπως φαίνεται παραπάνω στη δομή επανάληψης *Για ... από ... μέχρι* η αύξηση του μετρητή της επανάληψης δεν είναι ορατή, όπως συμβαίνει στις άλλες δύο επαναληπτικές δομές. Αυτό αποτελεί μια πηγή δυσκολιών και παρανοήσεων των αρχάριων προγραμματιστών (Cetin, 2015· du Boulay, 1986· Putnam et al., 1986· Yamashita, et al.. 2016) και θα μας απασχολήσει σε αυτή τη διατριβή δεδομένου ότι σε αλγόριθμους επεξεργασίας πινάκων η μεταβλητή-μετρητής μιας επανάληψης έχει διττό ρόλο αφού αποτελεί και δείκτη στα στοιχεία του πίνακα.

Η οικοδόμηση της επαναληπτικής διαδικασίας συνίσταται σε τρία βασικά σημεία (Κόμης, 2005):

- Οικοδόμηση και σχεδιασμός της επεξεργασίας, δηλαδή διατύπωση των εντολών που θα επαναληφθούν. Αυτό προϋποθέτει την ικανότητα γενίκευσης διάφορων μπλοκ εντολών που υλοποιούν την ίδια λειτουργία σε διαφορετικά δεδομένα.
- Προσδιορισμός της αρχικής κατάστασης των μεταβλητών του βρόχου.
- Προσδιορισμός της συνθήκης τερματισμού.

Ειδικά οι εμφωλευμένοι βρόχοι δημιουργούν πολλά προβλήματα στους αρχάριους προγραμματιστές (Cetin, 2015· Yamashita et. al., 2016), επειδή η εξοικείωση με αυτούς προϋποθέτει την πλήρη κατανόηση των παραπάνω σημείων. Επίσης πολλές φορές οι αρχάριοι προγραμματιστές δυσκολεύονται να παρακολουθήσουν τη ροή των εντολών μιας επαναληπτικής δομής (Yamashita et. al., 2016).

### *2.7.8 Δυσκολίες-Παρανοήσεις των μαθητών στην αναδρομή*

Η αναδρομή αποτελεί κεντρική έννοια της επιστήμης της πληροφορικής, αφού παίζει σημαντικό ρόλο στη θεωρητική θεμελίωσή της. Επίσης η δυνατότητα που έχει για την επίλυση πολύπλοκων προγραμματιστικών προβλημάτων με απλούστερο τρόπο σε σύγκριση με τις επαναληπτικές δομές, της δίνει δεσπόζουσα θέση στα μαθήματα προγραμματισμού και αλγορίθμων. Ένας αναδρομικός αλγόριθμος είναι συχνά ευκολότερος στην επινόηση και τη διατύπωση από τον αντίστοιχο επαναληπτικό (Rinderknecht, 2014).

Ωστόσο η χρήση της αναδρομής παρουσιάζει αρκετά διδακτικά προβλήματα. Οι δυσκολίες που αντιμετωπίζουν οι αρχάριοι προγραμματιστές με την αναδρομή σύμφωνα με πολλούς ερευνητές δεν έχουν να κάνουν τόσο με τη φύση της αναδρομής αλλά κυρίως με το γεγονός η κατανόησή της προϋποθέτει την εξοικείωση με κάποιες προγραμματιστικές δομές και μηχανισμούς (Velazquez-Iturbide, 2000). Εκτός από τα υποπρογράμματα και τον μηχανισμό μεταβίβασης παραμέτρων, σημαντικό ρόλο παίζει και ο τρόπος με τον οποίο λειτουργεί η στοίβα εκτέλεσης μέσω της οποίας υλοποιείται η αναδρομή. Όλες αυτές οι έννοιες είναι αρκετά σύνθετες για ένα εισαγωγικό μάθημα προγραμματισμού.

Μια λύση για το πρόβλημα αυτό είναι η παράκαμψη αυτών των εννοιών διδάσκοντας στους φοιτητές συναρτησιακό και όχι προστακτικό προγραμματισμό. Αρκετοί ερευνητές (Joy & Matthews, 2006· Joosten, van den Berg & van der Hoeven, 1993· Vandenberg & Wollowski, 2000) υποστηρίζουν αυτή τη στρατηγική με το επιχείρημα ότι στον συναρτησιακό προγραμματισμό οι επαναληπτικές διαδικασίες υλοποιούνται με χρήση της αναδρομής με αποτέλεσμα η κωδικοποίηση πολύπλοκων προβλημάτων να είναι πολύ πιο απλή.

Ωστόσο τα αποτελέσματα των ερευνών που έχουν γίνει δεν συμφωνούν ως προς το ποια είναι η καλύτερη προσέγγιση για ένα εισαγωγικό μάθημα προγραμματισμού. Σε μια από τις πρώτες έρευνες που έγιναν (Kessler & Anderson, 1986) τα αποτελέσματα έδειξαν ότι οι φοιτητές που ξεκίνησαν με αναδρομή δυσκολεύτηκαν πολύ περισσότερο στη μετάβαση στον προστακτικό προγραμματισμό από τους φοιτητές που ακολούθησαν την αντίστροφη πορεία. Η έρευνα καταλήγει στο συμπέρασμα ότι ο προστακτικός προγραμματισμός αποτελεί βασική προϋπόθεση για να κατανοήσει κάποιος την έννοια της αναδρομής.

Είναι φανερό ότι η κατανομή των φοιτητών όπως και το είδος των προβλημάτων που καλούνται να αντιμετωπίσουν παίζουν σημαντικό ρόλο σε αυτές τις έρευνες. Για παράδειγμα οι McCauley, Hanks, Fitzgerald, & Murphy (2015) επανέλαβαν την ίδια ακριβώς έρευνα με τους Benander, Benander, & Howard (1996) 19 χρόνια μετά καταλήγοντας σε αντίθετα αποτελέσματα.

### *2.7.9 Δυσκολίες-Παρανοήσεις των μαθητών στις δομές δεδομένων*

Στη βιβλιογραφία υπάρχουν διάφορες έρευνες για τις δυσκολίες των αρχάριων προγραμματιστών σε διάφορες δομές δεδομένων. Η ομάδα των Φιλανδών Seppällä, Malmi & Korhonen (2006) έχει μελετήσει τις παρανοήσεις των φοιτητών πάνω στον αλγόριθμο κατασκευής της δομής του σωρού (heap) μέσα από ασκήσεις οπτικοποιημένων προσομοιώσεων αλγορίθμων. Σε πιο πρόσφατες έρευνες (Danielsiek et al., 2012· Karpierz & Wolfman, 2014) οι επιστήμονες ταυτοποιούν και διερευνούν τις παρανοήσεις των φοιτητών στη δομή του σωρού, στα δυαδικά δέντρα και στους πίνακες κατακερματισμού.

Σε μια πιο πρόσφατη έρευνα (Grissom et al., 2016) διερευνήθηκαν οι δυσκολίες που αντιμετωπίζουν οι φοιτητές όταν σχεδιάζουν αναδρομικούς αλγόριθμους για την επεξεργασία δυαδικών δέντρων αναζήτησης. Ακόμα και οι φοιτητές που είχαν πρόσβαση σε κάποιο περιβάλλον προγραμματισμού και μπορούσαν να δοκιμάσουν και να διορθώσουν τον αλγόριθμο που είχαν σχεδιάσει υπέπεσαν σε αρκετά λάθη. Αρκετά από τα λάθη αυτά δεν οφείλονταν μόνο στη δομή δεδομένων αλλά και στη σωστή διατύπωση της αναδρομής.

Ωστόσο μόνο ένα πολύ μικρό ποσοστό της δουλειάς που έχει γίνει αναφέρεται σε παρανοήσεις των μαθητών ή φοιτητών στη δομή του πίνακα.

### *2.7.10 Δυσκολίες-Παρανοήσεις των μαθητών στους πίνακες*

Ο πίνακας αποτελεί μια από τις θεμελιώδεις δομές δεδομένων της πληροφορικής και σε πολλές περιπτώσεις χρησιμοποιείται για την υλοποίηση πιο σύνθετων δομών δεδομένων όπως είναι οι στοίβα, η ουρά, το δέντρο και ο γράφος. Σύμφωνα με μια μεγάλη έρευνα που έγινε σε εκπαιδευτικούς πληροφορικής (Dale, 2006), ο πίνακας αποτελεί μια από τις δυσκολότερες έννοιες σε ένα εισαγωγικό μάθημα πληροφορικής για τους μαθητές. Παρόλα αυτά δεν υπάρχει κάποια έρευνα η οποία να μελετά αποκλειστικά τις δυσκολίες των μαθητών στον προγραμματισμό με πίνακες.

Σε μια από τις πρώτες έρευνες για τις δυσκολίες που αντιμετωπίζουν οι μαθητές στον προγραμματισμό, ο du Boulay (1986) αναφέρει ότι οι μαθητές δυσκολεύονται να διακρίνουν τη διαφορά μεταξύ της έννοιας του δείκτη/θέσης/αναφοράς ενός στοιχείου του πίνακα και της τιμής του στοιχείου αυτού π.χ. 3, A[3]. Το ίδιο πρόβλημα αναφέρεται και σε μια άλλη έρευνα (Lister et al., 2006) όπου οι συγγραφείς μελετούν τις απαντήσεις των μαθητών σε διάφορα προβλήματα χρησιμοποιώντας την ταξινομία SOLO, που χρησιμοποιείται σε αυτή τη διατριβή. Το πρόβλημα αυτό είναι πιο έντονο όταν στη θέση του δείκτη υπάρχει μεταβλητή ή αριθμητική έκφραση (A[2*i] ) σε σχέση με τις περιπτώσεις που ο δείκτης είναι ένας αριθμός.

Τα ίδια και μεγαλύτερα προβλήματα παρουσιάζουν οι αρχάριοι προγραμματιστές με τους δείκτες (pointers) στη μνήμη που χρησιμοποιούν γλώσσες όπως η C (Craig & Petersen, 2016· Adcock et al., 2007). Φαίνεται, λοιπόν ότι η έννοια του δείκτη ως αναφορά σε στοιχείο του πίνακα ή άλλο στοιχείο δυσκολεύει αρκετά τους μαθητές.

Ένα χαρακτηριστικό παράδειγμα το οποίο καταδεικνύει τις δυσκολίες των μαθητών με τους πίνακες είναι το παρακάτω όπου η τιμή A[3] αποτελεί αναφορά σε ένα άλλο στοιχείο του πίνακα.

$$A[3] \leftarrow A[3] + A[5]$$

$$A[4] \leftarrow A[A[3]] + 5$$

Η δεύτερη ανάθεση τιμής μπορεί να προκαλέσει σύγχυση και ερωτήματα στους μαθητές, διότι η τιμή του στοιχείου του πίνακα παίζει ταυτόχρονα και το ρόλο του δείκτη σε ένα άλλο στοιχείο. Από αυτό φαίνεται ότι οι μαθητές ενδεχομένως δεν έχουν αντιληφθεί ότι το στοιχείο ενός πίνακα είναι ουσιαστικά μια μεταβλητή και μπορεί να χρησιμοποιηθεί όπως μια μεταβλητή. Η συγκεκριμένη παρανόηση προέκυψε από την εμπειρία μας μέσα στην τάξη και επιβεβαιώνεται σε αυτή την διατριβή μέσα από μια μεγάλη έρευνα που έγινε σε μαθητές 3 γενικών λυκείων (Βραχνός & Τζιμογιάννης, 2010· Vrachnos & Jimoyiannis, 2017).

Η έννοια του πίνακα οικοδομείται πάνω στην έννοια της μεταβλητής, αφού ένας πίνακας αποτελείται ουσιαστικά από ένα σύνολο μεταβλητών του ίδιου τύπου. Η διαφορά έγκειται στην ομαδοποιημένη διαχείριση των μεταβλητών αυτών με χρήση δομών επανάληψης, πράγμα που δεν μπορεί να γίνει με μεμονωμένες μεταβλητές. Κατά συνέπεια, μπορούμε να υποθέσουμε ότι οι αναπαραστάσεις και οι παρανοήσεις των μαθητών για την έννοια της μεταβλητής μεταφέρονται και στην έννοια του πίνακα.

Η οπτική αναπαράσταση που χρησιμοποιείται συνήθως για έναν πίνακα τόσο στα σχολικά εγχειρίδια όσο και στη διδασκαλία, είναι αυτή της ακολουθίας διαδοχικών κελιών. Είναι κυρίως στατική και, κατά συνέπεια, δεν επιτρέπει να αναδειχθούν τα δυναμικά χαρακτηριστικά του πίνακα που είναι απαραίτητα για την κατανόηση και την εφαρμογή βασικών αλγορίθμων επεξεργασίας (αναζήτηση, ταξινόμηση κ.α.).

### *2.7.11 Δυσκολίες των μαθητών στους αλγόριθμους ταξινόμησης*

Ένα άλλο σημαντικό θέμα το οποίο, επίσης, έχει μελετηθεί ελάχιστα στη βιβλιογραφία (Geller & Dios, 1998· Simon et al., 2006) είναι οι δυσκολίες που παρουσιάζουν οι μαθητές στην κατανόηση της λειτουργίας των αλγορίθμων ταξινόμησης και αναζήτησης σε πίνακες. Οι αλγόριθμοι επεξεργασίας πινάκων είναι δύσκολο να παρουσιαστούν στους μαθητές με παραδοσιακά μέσα (πίνακας, μολύβι, χαρτί), λόγω της εξαιρετικά δυναμικής φύσης τους. Οι αλγόριθμοι αυτοί συνίστανται συνήθως σε συγκρίσεις, μετακινήσεις και αντιμεταθέσεις στοιχείων. Αρκετοί εκπαιδευτικοί χρησιμοποιούν πλέον λογισμικά οπτικοποίησης για την παρουσίαση των αλγόριθμων αυτών με επεξηγηματικό τρόπο στους μαθητές.

Η ταξινόμηση αποτελεί ένα από τα θεμελιώδη αλγοριθμικά προβλήματα και χρησιμοποιείται για την εισαγωγή μαθητών ή φοιτητών σε τεχνικές σχεδίασης και ανάλυσης αλγορίθμων. Παρότι υπάρχει μεγάλη ποικιλία αλγόριθμων ταξινόμησης, ο αλγόριθμος ευθείας ανταλλαγής (straight exchange sort), που είναι ευρύτερα γνωστός και ως αλγόριθμος ταξινόμησης φυσαλίδας (bubble sort), είναι από τους πιο δημοφιλείς αλγόριθμους ταξινόμησης στην εκπαίδευση. Βασίζεται στην σύγκριση και αντιμετάθεση γειτονικών στοιχείων ενός πίνακα μέχρις ότου τα στοιχεία του πίνακα να διαταχθούν στην επιθυμητή σειρά. Διδάσκεται δε, στο πλαίσιο του μαθημάτων α)Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον της κατεύθυνσης Οικονομίας – Πληροφορικής της Γ’ Λυκείου και β) Προγραμματισμός Υπολογιστών της Γ’ τάξης του τομέα πληροφορικής του ΕΠΑΛ.

Ωστόσο ο αλγόριθμος αυτός δυσκολεύει αρκετά τους μαθητές, κάτι που επιβεβαιώθηκε από την αντίστοιχη έρευνα που εκπονήθηκε στο πλαίσιο της διατριβής (Βραχνός & Τζιμογιάννης, 2014α). Πρόκειται για την πρώτη φορά που οι μαθητές έρχονται αντιμέτωποι με έναν αλγόριθμο επεξεργασίας πινάκων με εμφωλευμένη επανάληψη ο οποίος αναφέρεται στο ίδιο βήμα σε δύο διαφορετικά στοιχεία (A[j], A[j–1]). Επίσης η ταξινόμηση ευθείας ανταλλαγής αποτελεί έναν αλγόριθμο ο οποίος δεν προκύπτει άμεσα από την εμπειρία των μαθητών και δεν μπορεί εύκολα να χτιστεί σε

προϋπάρχουσες γνώσεις ή εμπειρίες από την καθημερινή ζωή (Βραχνός & Τζιμογιάννης, 2016· 2018), αφού σχεδόν κανείς δεν χρησιμοποιεί αυθόρμητα τον αλγόριθμο φυσαλίδας για να ταξινομήσει ένα σύνολο αντικειμένων με το χέρι.

Επίσης, από διδακτική σκοπιά, δεν μπορεί να προσεγγιστεί εύκολα με πρακτικές διερευνητικής μάθησης και εποικοδόμησης πάνω στην προϋπάρχουσα αλγοριθμική γνώση των μαθητών, όπως μπορεί να γίνει με τον αλγόριθμο ταξινόμησης επιλογής (Βραχνός & Τζιμογιάννης, 2016· 2018). Αυτοί είναι κάποιοι από τους λόγους που οι μαθητές δυσκολεύονται πολύ στην κατανόηση της λειτουργίας αυτού του αλγορίθμου. Αντίθετα οι αλγόριθμοι ταξινόμησης επιλογής (selection sort) και εισαγωγής (insertion sort) είναι πιο κοντά στην καθημερινή πρακτική και τις προϋπάρχουσες γνώσεις των μαθητών (Βραχνός & Τζιμογιάννης, 2016).

Ωστόσο ακόμα και αυτοί οι αλγόριθμοι βασίζονται σε προγραμματιστικές δομές οι οποίες είναι δύσκολο να οικοδομηθούν με τα παραδοσιακά διδακτικά μέσα κυρίως λόγω της δυναμικής φύσης τους, αφού χαρακτηρίζονται από τη διαρκή τροποποίηση της δομής τους μέσω εναλλαγής και μετακίνησης στοιχείων.

## *2.8 Η ταξινομία γνωστικών στόχων του Bloom στον προγραμματισμό*

Η σπουδαιότητα της αξιολόγησης των προσδοκώμενων αποτελεσμάτων στην εκπαιδευτική διαδικασία έχει οδηγήσει τους επιστήμονες της εκπαίδευσης σε πολλές προσπάθειες ταξινόμησής τους. Για αυτό έχουν προταθεί διάφορες ταξινομίες προσδοκώμενων αποτελεσμάτων ή επίτευξης των διδακτικών στόχων που θέτουμε. Σκοπός των ερευνητών και των εκπαιδευτικών είναι να επιτευχθεί η καλύτερη δυνατή ανάλυση της αλγοριθμικής σκέψης των μαθητών και της γνώσης που απαιτείται για την επίλυση προβλημάτων. Για αυτόν τον σκοπό γίνεται προσπάθεια σχεδιασμού εργαλείων αξιολόγησης της απόδοσης των μαθητών στην επίλυση προβλημάτων που επιδέχονται αλγοριθμική λύση. Οι ταξινομίες που χρησιμοποιούνται προσπαθούν να κατατάξουν σε επίπεδα τα αποτελέσματα με βάση τον βαθμό της ποιότητας, της πολυπλοκότητας και της αφαίρεσης.

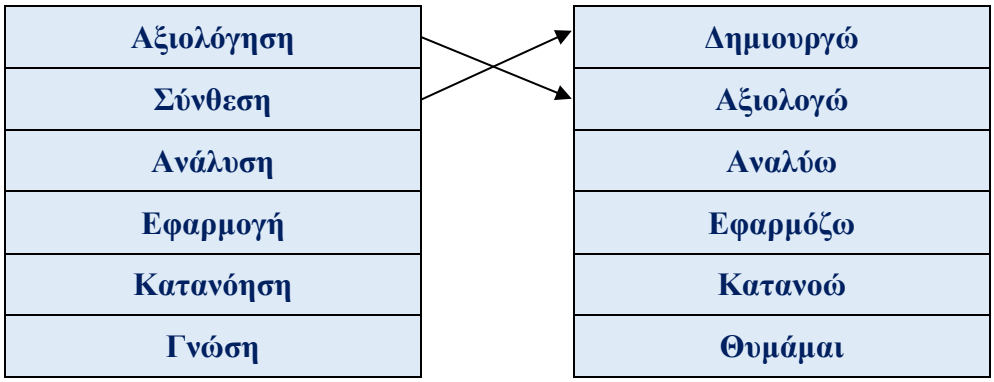
Το πιο διαδεδομένο σχήμα ταξινόμησης στόχων είναι γνωστό ως ταξινομία Bloom (1956). Το σχήμα αυτό κατατάσσει τα προσδοκώμενα αποτελέσματα σε τρεις μεγάλους τομείς: τον γνωστικό, τον συναισθηματικό και τον ψυχοκινητικό. Σε αυτή τη διατριβή επικεντρωνόμαστε στον γνωστικό τομέα ο οποίος αναλύεται σε πέντε επίπεδα τα οποία

δίνονται παρακάτω σε αύξουσα σειρά βαθμού πολυπλοκότητας και παιδαγωγικής σπουδαιότητας:

- *Γνώση* : Ο μαθητής απομνημονεύει και είναι σε θέση να ανακαλέσει γεγονότα, θεωρίες κλπ.
- *Κατανόηση*: Ο μαθητής έχει πλήρη συνείδηση όσων στοιχείων απομνημονεύει και μπορεί να τα μετασχηματίσει σε πιο κατανοητές μορφές. Μπορεί να περιγράψει ένα αντικείμενο με δικά του λόγια.
- *Εφαρμογή*: Ο μαθητής μπορεί να χρησιμοποιήσει τη γνώση που απέκτησε για να επιλύσει διάφορα προβλήματα.
- *Ανάλυση*: Ο μαθητής μπορεί να διακρίνει τα συστατικά μέρη ενός αντικειμένου και τον τρόπο σύνδεσής τους.
- *Σύνθεση* : Ο μαθητής μπορεί να σχεδιάσει και να κατασκευάσει μια νέα δομή από επιμέρους τμήματα.
- *Αξιολόγηση*: Ο μαθητής μπορεί να διατυπώσει αξιολογικές κρίσεις και να λαμβάνει αποφάσεις για μεθόδους και ιδέες.

Για να μπορέσει ένας μαθητής να φτάσει σε ένα επίπεδο θα πρέπει πρώτα να έχει κατακτήσει όλα τα προηγούμενα. Το 2001 προτάθηκε από τους Anderson & Krathwohl (2001) μια αναθεωρημένη έκδοση της ταξινομίας του Bloom στην οποία τα ονόματα των κατηγοριών από ουσιαστικά έγιναν ρήματα για να δείξουν τον ενεργό ρόλο του μαθητή. Η σημαντικότερη αλλαγή είναι η αντιμετάθεση των δύο υψηλότερων επιπέδων. Ενώ η αξιολόγηση ήταν στο υψηλότερο επίπεδο έπεσε μια θέση δίνοντας τη θέση της στην δημιουργία που αντικατέστησε τη σύνθεση. Επίσης όσον αφορά το επίπεδο της κατανόησης υπάρχει μια διαφορά που δεν φαίνεται στην ελληνική μετάφραση. Στην αναθεωρημένη ταξινομία Bloom η λέξη understanding έχει αντικατασταθεί από το ρήμα comprehend το οποίο αναφέρεται σε βαθιά κατανόηση (thorough understanding).

Ωστόσο, η εφαρμογή και η ερμηνεία των παραπάνω ταξινομιών στον προγραμματισμό υπολογιστών φαίνεται να είναι προβληματική (Shuhidan, Hamilton & D' Souza, 2009· Thompson et al. 2008) αφού σε πολλές περιπτώσεις η αντιστοίχιση των λύσεων των μαθητών με τα επίπεδα της ταξινομίας δεν είναι άμεση (Fuller et al. 2007· Thomson et al. 2008).




**Σχήμα 2.1. Γνωστικά επίπεδα ταξινομίας Bloom και αναθεωρημένης Bloom**

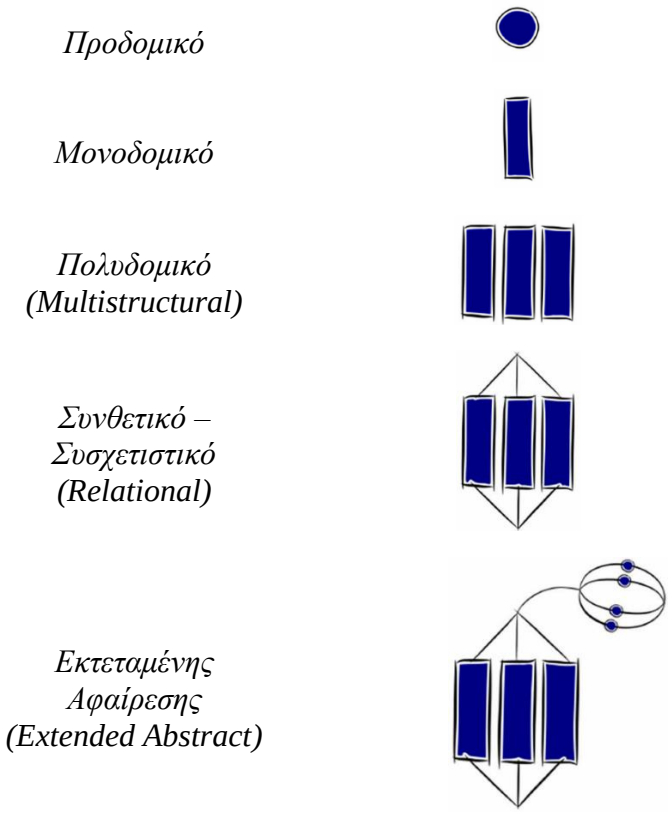
Ένα άλλο πρόβλημα σχετίζεται με την εγγενή δυσκολία της ακριβής κατάταξης ενός τμήματος κώδικα σε κάποιο γνωστικό επίπεδο, λόγω των διαφορετικών απόψεων για την πολυπλοκότητα του κώδικα και των γνωστικών διαδικασιών που απαιτούνται για την επίλυση του υπό μελέτη προβλήματος (Whalley & Kasto, 2014).

## *2.9 Η ταξινομία SOLO στον προγραμματισμό*

Υιοθετώντας μια πιο εποικοδομιστική προσέγγιση, οι Biggs & Collis (1982) πρότειναν την ταξινομία SOLO (**S**tructure of **O**bserved **L**earning **O**utcomes) για να περιγράψουν μια ιεραρχία κατηγοριών-επιπέδων, τα οποία εκφράζουν την εξελικτική πορεία της διαδικασίας οικοδόμησης της γνώσης. Στην περίπτωση του προγραμματισμού, η εξελικτική αυτή διαδικασία μπορεί να αφορά την συνεχή βελτίωση ενός τμήματος κώδικα, ανεβαίνοντας παράλληλα τα γνωστικά επίπεδα της ταξινομίας. Σύμφωνα με την ταξινομία SOLO η μαθησιακή διαδικασία συντελείται σε μια σειρά από στάδια αυξανόμενης πολυπλοκότητας τα οποία δεν έχουν απαραίτητα κάποια λογική δομή, αλλά κάθε στάδιο βρίσκεται σε ένα υψηλότερο επίπεδο αφαίρεσης από το προηγούμενο.

## *2.10 Εφαρμογή της SOLO στην εκπαιδευτική έρευνα*

Η ταξινομία SOLO περιλαμβάνει πέντε επίπεδα δομικής πολυπλοκότητας της γνώσης των μαθητών.




**Σχήμα 2.2. Γνωστικά επίπεδα της ταξινομίας SOLO**

Στο **προδομικό** (*prestructural*) επίπεδο κατατάσσονται οι απαντήσεις που δεν σχετίζονται με το πρόβλημα ή έχουν βασικά λάθη σε προγραμματιστικές δομές.

Στο επόμενο επίπεδο της ταξινομίας το **μονοδομικό** (*unistructural*) κατηγοριοποιούνται απαντήσεις, στις οποίες ο μαθητής εξηγεί τι κάνει ένα τμήμα εντολών γραμμή προς γραμμή, αλλά δεν έχει οικοδομήσει λειτουργικές αναπαραστάσεις για ολόκληρο το τμήμα κώδικα που του δίνεται.

Στο **πολυδομικό** (*multistructural*) επίπεδο κατατάσσονται απαντήσεις παρόμοιες με αυτές του μονοδομικού, μόνο που τώρα αφορούν όλο το τμήμα του κώδικα. Δηλαδή ο μαθητής αντιλαμβάνεται τη λειτουργία κάθε εντολής μεμονωμένα και μπορεί να εκτελέσει τον αλγόριθμο στο χαρτί βήμα-βήμα. Δεν είναι όμως σε θέση να οικοδομήσει ολοκληρωμένες αναπαραστάσεις για το συνολικό ρόλο ενός τμήματος εντολών αλγορίθμου και, κατά συνέπεια, για το σκοπό που σχεδιάστηκε.

Στο **συνθετικό/συσχετιστικό** (relational) επίπεδο ο μαθητής μπορεί να διακρίνει την λειτουργία που επιτελεί ο αλγόριθμος σε ένα πιο αφηρημένο επίπεδο.

Το ανώτερο επίπεδο της ταξινομίας είναι το επίπεδο **θεωρητικής γενίκευσης ή εκτεταμένης αφαίρεσης** (extended abstract) στο οποίο ο μαθητής συνδέει το

πρόβλημα με ένα ευρύτερο πλαίσιο. Δηλαδή έχει την ικανότητα να γενικεύσει το πρόβλημα αξιοποιώντας στρατηγικές γνώσεις και δεξιότητες προγραμματισμού που έχει αναπτύξει. Το επίπεδο αυτό καταγράφεται σπάνια σε έννοιες και εργαλεία που χρησιμοποιούνται στα εισαγωγικά μαθήματα προγραμματισμού, και δεν θα μας απασχολήσει σε αυτή την εργασία.

Όσο ανεβαίνουμε επίπεδο προχωράμε σε υψηλότερο βαθμό αφαίρεσης. Σε χαμηλό επίπεδο θεωρούμε ότι η κατανόηση είναι επιφανειακή ενώ όσο προχωράμε γίνεται πιο βαθιά από εννοιολογικής άποψης.

Η ταξινομία SOLO μπορεί να χρησιμοποιηθεί για να ορίσει προσδοκώμενα μαθησιακά αποτελέσματα και τρόπους αξιολόγησης του βαθμού επίτευξης τους. Έχει χρησιμοποιηθεί σε διάφορα αντικείμενα όπως είναι τα μαθηματικά (Chick, 1998), η βιολογία (Campbell et al., 1998), οι γλωσσικές σπουδές (Lake, 1999), η ποίηση και η ιστορία (Biggs & Collis, 1982).

Η ομάδα του Lister (Lister et al., 2006), χρησιμοποίησε για πρώτη φορά την ταξινομία SOLO για την ανάλυση των απαντήσεων φοιτητών σε προβλήματα προγραμματισμού. Στη συνέχεια χρησιμοποιήθηκε ευρέως από πολλούς ερευνητές για την ανάλυση των απαντήσεων των φοιτητών διάφορες κατηγορίες προγραμματιστικών προβλημάτων (Clear et al., 2008· Corney et al., 2014· Seiter, 2015· Sheard et al., 2008· Whalley et al., 2011).

Πιο πρόσφατα οι Meerbaum-Salant, Armoni & Ben-Ari (2013) πρότειναν έναν συνδυασμό της αναθεωρημένης ταξινομίας Bloom και της SOLO για τον σχεδιασμό προγραμματιστικών δραστηριοτήτων στο Scratch. Επίσης οι Ginat & Menashe (2015) παρουσίασαν ένα πλαίσιο για την εφαρμογή της SOLO στην αξιολόγηση αλγοριθμικών λύσεων ενώ ο Seiter (2015) χρησιμοποίησε τη SOLO για να εκτιμήσει τις δεξιότητες υπολογιστικής σκέψης μαθητών της τετάρτης δημοτικού.

Τα αποτελέσματα ανεξαρτήτων ερευνών (Clear et al. 2008· Sheard et al., 2008· Thompson, 2007) έδειξαν ότι η SOLO μπορεί να χρησιμοποιηθεί αποτελεσματικά είτε για την αξιολόγηση των απαντήσεων των μαθητών σε προβλήματα ανάγνωσης κώδικα είτε για την αξιολόγηση των προγραμμάτων που συντάσσουν οι μαθητές για την επίλυση προβλημάτων. Κατά την αξιολόγηση των απαντήσεων των μαθητών πρέπει να δοθεί ιδιαίτερη προσοχή, σε περίπτωση που τα προβλήματα που έχουν τεθεί είναι γνωστά στους μαθητές. Για παράδειγμα έστω ότι δύο μαθητές δίνουν ακριβώς την ίδια απάντηση, όμως ο ένας κάνει απλά ανάκληση της λύσης που γνωρίζει ενώ ο άλλος

σχεδιάζει τη λύση εκείνη τη στιγμή. Προφανώς οι δύο αυτές απαντήσεις δεν ανήκουν στο ίδιο επίπεδο της ταξινομίας. Άρα η κατάταξη των απαντήσεων των μαθητών στα επίπεδα της ταξινομίας SOLO εξαρτάται και από τις προϋπάρχουσες γνώσεις τους (Izu & Weerasinghe, 2016· Petersen, Graig & Zingaro, 2011).

Στη συνέχεια περιγράφουμε τα επίπεδα της ταξινομίας SOLO αναλυτικά μέσα από δύο παραδείγματα που δίνονται στο Τμήμα Κώδικα 2.5.

```
Αύξουσα ← Αληθής
Για j από 2 μέχρι N
    Αν array[ j ] < array[ j-1] Τότε
        Αύξουσα ← Ψευδής
    Τέλος_Αν
Τέλος_Επανάληψης
```

```
a[ j ] ← a[ j ] + 1
```

**Τμήμα Κώδικα 2.5. α) Έλεγχος αύξουσας διάταξης β) αύξηση μεταβλητής κατά 1**

Η ερμηνεία ότι το τμήμα κώδικα (α) *εκτελεί ταξινόμηση των στοιχείων του πίνακα* κατατάσσεται στο χαμηλότερο επίπεδο πολυπλοκότητας της ταξινομίας, το προδομικό. Η απάντηση αυτή δείχνει ότι ο μαθητής δεν έχει καμία κατανόηση έστω μεμονωμένων εντολών του κώδικα, αλλά προσπαθεί να θυμηθεί με ποιον γνωστό αλγόριθμο μοιάζει.

Παραδείγματα απαντήσεων που κατατάσσονται στο μονοδομικό επίπεδο για το τμήμα κώδικα (β) είναι η απάντηση "*Η εντολή εκχωρεί στο a[j] το a[j]+1*", ενώ για το (α) ότι "*ο αλγόριθμος συγκρίνει στοιχεία του πίνακα*" .

Επίσης, στην περίπτωση της εντολής a[j] ← a[j] + 1 μια απάντηση μαθητή που κατατάσσεται στο πολυδομικό επίπεδο είναι ότι "*η εντολή αυξάνει το i-οστό στοιχείο του πίνακα*". Η απάντηση αυτή είναι στην πραγματικότητα στο υψηλότερο επίπεδο αφαίρεσης του πολυδομικού και θα μπορούσε από κάποιους να θεωρηθεί ότι ανήκει στο επόμενο επίπεδο. Επίσης μια απάντηση για το τμήμα κώδικα (α) θα μπορούσε να είναι η εξής:

*Ο αλγόριθμος αρχικά θέτει στη μεταβλητή Αύξουσα την τιμή Αληθής και στη συνέχεια ελέγχει αν κάθε στοιχείο του πίνακα array[j] είναι μικρότερο του array[j-1]. Τότε εκχωρεί στη μεταβλητή Αύξουσα την τιμή Ψευδής αλλιώς δεν κάνει τίποτα.*

Η παραπάνω περιγραφή είναι μια κλασική περιγραφή τμήματος κώδικα που κατατάσσεται στο πολυδομικό επίπεδο. Μια άλλη περιγραφή η οποία δείχνει ένα υψηλότερο επίπεδο αφαίρεσης είναι η παρακάτω:

*Ο αλγόριθμος ελέγχει τα στοιχεία του πίνακα ανά ζεύγη και αν βρει δύο διαδοχικά στοιχεία τα οποία βρίσκονται σε φθίνουσα σειρά εκχωρεί στη μεταβλητή Αύξουσα την τιμή Ψευδής.*

Αυτή η περιγραφή βρίσκεται οριακά μεταξύ του πολυδομικού και του αμέσως επόμενου επιπέδου του συνθετικού.

Στο παράδειγμα (α) μια συνθετική απάντηση είναι ότι "*ελέγχει αν τα στοιχεία ενός πίνακα είναι σε αύξουσα σειρά*" ενώ για το (β) "*το στοιχείο a[j] του πίνακα παίζει το ρόλο ενός μετρητή*". Εδώ ο μαθητής έχει κατανοήσει τον γενικότερο σκοπό για τον οποίο έχει σχεδιαστεί ο αλγόριθμος.

Τέλος όσον αφορά το επίπεδο θεωρητικής γενίκευσης. στο παράδειγμα (α) θα μπορούσε κάποιος μαθητής να διακρίνει, ότι ο αλγόριθμος εκτελεί περιττές συγκρίσεις σε περίπτωση που διαπιστώσει νωρίς ότι ο πίνακας δεν είναι σε αύξουσα σειρά. Ο αλγόριθμος μπορεί να τροποποιηθεί έτσι ώστε να διακόπτει τις συγκρίσεις μόλις βρει δύο διαδοχικά στοιχεία που δεν είναι σε αύξουσα σειρά, όπως φαίνεται στο Τμήμα Κώδικα 2.6.

```
Αύξουσα ← Αληθής
i ← 2
Όσο i ≤ N και Αύξουσα Επανάλαβε
    Αν array[ j ] < array[ j-1] Τότε
        Αύξουσα ← Ψευδής
    Τέλος_Αν
    i ← i + 1
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 2.6. Έλεγχος αύξουσας διάταξης χωρίς περιττές επαναλήψεις**

Σε αυτή την περίπτωση ο μαθητής όχι μόνο έχει κατανοήσει τη λειτουργία του αλγορίθμου στο υψηλότερο επίπεδο αφαίρεσης, αλλά βελτίωσε τον αλγόριθμο έτσι ώστε να έχει καλύτερη απόδοση.

Ένα ενδιαφέρον χαρακτηριστικό της ταξινομίας SOLO είναι ότι μια σωστή απάντηση μπορεί να καταταχθεί σε χαμηλότερο επίπεδο από μια λάθος απάντηση. Αυτό μπορεί να συμβεί στην περίπτωση που η λύση του μαθητή περιέχει κάποιο όχι σημαντικό συντακτικό ή λογικό λάθος αλλά η κεντρική ιδέα παραπέμπει σε υψηλό βαθμό αφαίρεσης. Χαρακτηριστική είναι η έκφραση που χρησιμοποιεί ο Lister (Lister et al. 2006) ότι στα χαμηλά επίπεδα της ταξινομίας "*οι μαθητές χάνουν το δάσος και*

*βλέπουν μόνο τα δέντρα*". Δεν μας ενδιαφέρει δηλαδή τόσο πολύ αν το τελικό πρόγραμμα εκτελείται σωστά ή όχι, αλλά αν η απάντηση περιγράφει τη γενική ιδέα του αλγορίθμου σε υψηλό επίπεδο αφαίρεσης.

Αρκετοί ερευνητές (Corney et al., 2011· Lister et al., 2006) που ασχολήθηκαν με την εφαρμογή της ταξινομίας SOLO στον προγραμματισμό, ισχυρίζονται ότι οι μαθητές που δεν μπορούν να διαβάσουν και να περιγράψουν ένα τμήμα κώδικα στο συσχετιστικό επίπεδο της ταξινομίας, δεν έχουν αναπτύξει τις δεξιότητες που χρειάζονται ώστε να σχεδιάζουν δικά τους προγράμματα για την επίλυση προβλημάτων. Αν εξετάσουμε αυτόν τον ισχυρισμό μέσα από το πρίσμα της θεωρίας του εποικοδομισμού καταλήγουμε στο συμπέρασμα ότι το συσχετιστικό επίπεδο είναι πολύ σημαντικό για την ανάπτυξη των προγραμματιστικών δεξιοτήτων των μαθητών.

### *2.10.1 Μελέτη περίπτωσης*

Στη συνέχεια με τη βοήθεια του τμήματος κώδικα 2.7 το οποίο περιγράφει τον αλγόριθμο ταξινόμησης της ευθείας ανταλλαγής θα δώσουμε ένα σχήμα ανάλυσης της ταξινομίας SOLO επικεντρωμένο σε αλγορίθμους με πίνακες.

```
Για i από 2 μέχρι N
    Για j από N μέχρι i με βήμα -1
        Αν array[ j ] < array[ j-1] Τότε
            Αντιμετάθεσε array[ j ], array[ j-1]
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 2.7. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής**

**Πίνακας 2.1. Ενδεικτικές περιγραφές και παραδείγματα των επιπέδων της SOLO**

| **Επίπεδο SOLO**: προδομικό | |
|---|---|
| **Περιγραφή:**<br>• έχει μικρή η καμία σχέση με το ζητούμενο<br>• αντιγραφή / επανάληψη / απομνημόνευση<br>• Στοιχειώδεις ελλείψεις ή παρανοήσεις σε βασικά στοιχεία προγραμματισμού | **Ενδεικτικές απαντήσεις :**<br>• Ο αλγόριθμος αναζητά μια τιμή στον πίνακα<br>• Ο αλγόριθμος βρίσκει την μέγιστη τιμή στον πίνακα<br>• Ο μαθητής δεν μπορεί να διαχωρίσει την έννοια του δείκτη ενός πίνακα από τα περιεχόμενά του |

| **Επίπεδο SOLO**: μονοδομικό | |
|---|---|
| **Περιγραφή:**<br>• Επικεντρώνεται σε μεμονωμένη εντολή ή έννοια<br>• Περιγράφει τμήμα του κώδικα<br>• Ο μαθητής έχει κατανοήσει κάποιες αλλά όχι όλες τις πλευρές του προβλήματος | **Ενδεικτικές απαντήσεις :**<br>• Αν το στοιχείο array[j] είναι μικρότερο από το array[j-1] τότε του temp παίρνει την τιμή του array[j], στη συνέχεια το array[j] παίρνει την τιμή του array[j-1] και τέλος το array[j] παίρνει την τιμή του temp.<br>• Ο αλγόριθμος εκτελεί συγκρίσεις μεταξύ των στοιχείων του πίνακα |

| **Επίπεδο SOLO:** πολυδομικό | |
|---|---|
| **Περιγραφή:**<br>• Περιγράφει το μεγαλύτερο τμήμα του κώδικα γραμμή προς γραμμή χωρίς σημαντικές παραλείψεις ή λάθη.<br>• Περιγράφει το μεγαλύτερο τμήμα του κώδικα γραμμή προς γραμμή χωρίς να επικεντρώνεται στις συνδέσεις/σχέσεις μεταξύ των επιμέρους τμημάτων. | **Ενδεικτικές απαντήσεις :**<br>• Συγκρίνει κάθε στοιχείο του πίνακα με το επόμενο. Αν είναι μεγαλύτερο τότε τα δύο στοιχεία αλλάζουν θέσεις μεταξύ τους.<br>• Ο αλγόριθμος συγκρίνει γειτονικά στοιχεία. Αν δεν είναι σε αύξουσα σειρά τότε αλλάζουν θέση μεταξύ τους. |

| **Επίπεδο SOLO:** συνθετικό / συσχετιστικό | |
|---|---|
| **Περιγραφή:**<br>• Η απάντηση δείχνει κατανόηση του σκοπού και της λειτουργίας που επιτελεί ο κώδικας<br>• Η απάντηση δείχνει πολύ καλή σύνδεση και ενοποίηση των προγραμματιστικών δομών και αντικειμένων.<br>• Εφαρμογή της βασικής ιδέας ή της προγραμματιστικής έννοιας σε ένα πρόβλημα. | **Ενδεικτικές απαντήσεις :**<br>• Διατάσσει τα στοιχεία του πίνακα σε φθίνουσα σειρά<br>• Θέτει σε αύξουσα σειρά τα στοιχεία του πίνακα<br>• Ταξινομεί τα στοιχεία του πίνακα. |

| **Επίπεδο SOLO**: θεωρητικής γενίκευσης / εκτεταμένης αφαίρεσης |
|---|
| **Περιγραφή:**<br>• Αμφισβήτηση του υπάρχοντος προβλήματος και απάντηση πέρα από τα όρια του προβλήματος<br>• Χρήση των προγραμματιστικών δομών για την επίλυση πρωτότυπων και μην αναμενόμενων προβλημάτων<br>• Χρήση αλγοριθμικών δομών ή δομών δεδομένων που δεν απαιτούνται από το πρόβλημα, για τον σχεδιασμό μιας βέλτιστης λύσης |

### *2.10.2 Ενδεικτική απάντηση επιπέδου εκτεταμένης αφαίρεσης*

Το Τμήμα Κώδικα 2.8 ως απάντηση στο πρόβλημα που τέθηκε, αποτελεί έναν βελτιωμένο αλγόριθμο ο οποίος σταματάει αμέσως μόλις διαπιστωθεί ότι ο πίνακας έχει ταξινομηθεί χρησιμοποιώντας μια λογική μεταβλητή, για τον έλεγχο αυτό.·

Στην παραπάνω ανάλυση η απάντηση ότι ο αλγόριθμος εκτελεί φθίνουσα ταξινόμηση παρόλο που δεν είναι σωστή, κατατάσσεται στο συνθετικό επίπεδο αφού ο μαθητής έχει διακρίνει την λειτουργία του αλγορίθμου στο πιο υψηλό επίπεδο αφαίρεσης (εκτελεί ταξινόμηση). Η λανθασμένη εκτίμηση της διάταξης που επιφέρει η ταξινόμηση δεν είναι αρκετή ώστε να υποβιβάσει αυτή την απάντηση σε χαμηλότερο επίπεδο της ταξινομίας, διότι παραμένει στο ίδιο επίπεδο αφαιρετικής σκέψης.

```
i ← 2
Αρχή_Επανάληψης
    Ταξινομήθηκε ← Αληθής
    Για j από N μέχρι i με βήμα -1
        Αν array[ j ] < array[ j-1] Τότε
            Αντιμετάθεσε array[ j ], array[ j-1]
            Ταξινομήθηκε ← Ψευδής
        Τέλος_Αν
    Τέλος_Επανάληψης
    i ← i + 1
Μέχρις_ότου i > N ή Ταξινομήθηκε
```

**Τμήμα Κώδικα 2.8. Βελτιωμένος Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής**

Οι απαντήσεις που κατατάσσονται σε αυτό το επίπεδο δεν παρατηρούνται συχνά και είναι πολύ δύσκολο να δοθούν σε προβλήματα ανάγνωσης κώδικα όπου ζητείται από τους μαθητές να περιγράψουν τη λειτουργία ενός αλγορίθμου. Οι απαντήσεις αυτές παρατηρούνται συνήθως όταν ζητείται από τον μαθητή να σχεδιάσει έναν αλγόριθμο για την επίλυση ενός προβλήματος και ο μαθητής επινοεί μια πρωτότυπη λύση ή βελτιώνει μια ήδη υπάρχουσα, αξιοποιώντας δομές των οποίων η χρήση δεν προκύπτει άμεσα από το πρόβλημα ή κατασκευάζοντας νέες δομές οι οποίες διευκολύνουν την επίλυση του προβλήματος (Ginat & Menashe, 2015· Whalley & Kasto, 2014· Whalley et al., 2011).

Γενικά τέτοιες λύσεις συναντώνται σε πρωτότυπα προβλήματα που αντιμετωπίζουν πρώτη φορά οι μαθητές και πρέπει να επινοήσουν μια νέα λύση.

## *2.11 Εκπαιδευτικά περιβάλλοντα προγραμματισμού*

Για την αντιμετώπιση των παραπάνω δυσκολιών αλλά και για την εισαγωγή του προγραμματισμού σε μικρές ηλικίες έχουν αναπτυχθεί εκπαιδευτικά περιβάλλοντα ειδικά σχεδιασμένα για τη διδασκαλία του προγραμματισμού. Τα περισσότερα από αυτά τα περιβάλλοντα βασίζονται στη γεωμετρία της χελώνας μιας προσέγγισης που ήταν πολύ δημοφιλής τις προηγούμενες δεκαετίες μέσα από περιβάλλοντα χελωνόκοσμων (MSWLogo, MicroWolds Pro, KTurtle) όπου ο μαθητές προγραμμάτιζε στη γλώσσα Logo. Τα εκπαιδευτικά περιβάλλοντα που χρησιμοποιούνται σήμερα ενώ βασίζονται στην ίδια λογική, δίνουν τη δυνατότητα στους μαθητές να αναπτύσσουν ένα πρόγραμμα με οπτικό τρόπο, ενώνοντας μπλοκ

εντολών. Επίσης είναι εμπλουτισμένα με εντολές κίνησης, σχεδίασης γραφικών και μετάδοσης μηνυμάτων μεταξύ αντικειμένων τα οποία καθιστούν ελκυστική την ανάπτυξη γραφικών εφαρμογών και ηλεκτρονικών παιχνίδια από τους μαθητές. Περιβάλλοντα προγραμματισμού με πλακίδια (block based) όπως το Scratch (Resnick et al., 2009), το Snap! (Harvey & Mönig, 2010) και το Blockly (Fraser, 2015) χρησιμοποιούνται σήμερα για τη διδασκαλία του προγραμματισμού από το δημοτικό μέχρι το λύκειο. Υπάρχει ακόμα και μια έκδοση του Scratch to ScratchJr (Flannery et al., 2013) ειδικά σχεδιασμένο για παιδιά 5-7 ετών.

Με την αύξηση της δημοτικότητας αυτών και άλλων παρόμοιων εργαλείων, αυξάνεται όλο και περισσότερο ο αριθμός των δραστηριοτήτων με τις οποίες οι μαθητές μπορούν να ασχοληθούν με τον προγραμματισμό με πλακίδια (Bau et al., 2017). Έχουν τη δυνατότητα να αναπτύξουν εφαρμογές για κινητά με το App Inventor (Wolber et al., 2014), να δημιουργήσουν και να πειραματιστούν με υπολογιστικά μοντέλα με το DeltaTick (Wilkerson-Jerde, Wagh & Wilensky, 2015), το Frog Pond (Horn et al., 2014) και τη StarLogo TNG (Klopfer et al., 2009) ή να ζωγραφίζουν καλλιτεχνήματα με το Turtle Art (Bontá, Papert & Silverman, 2010). Επίσης μπορούν να αναπτύξουν δεξιότητες υπολογιστικής σκέψης μέσα από παιχνίδια όπως το code combat (Kumar, 2014) και το Lightbot (Gouws, Bradshaw & Wentworth, 2013). Πέρα από την ανάπτυξη πολυάριθμων εκπαιδευτικών περιβαλλόντων προγραμματισμού έχουν αναπτυχθεί πολλές δράσεις για τη διάδοση του προγραμματισμού σε παγκόσμια κλίμακα όπως είναι η ώρα του κώδικα (Du, Wimmer & Rada, 2018) και η πρωτοβουλία *made with code* ( https://www.madewithcode.com/ ) της Google.

Τα τελευταία χρόνια υπάρχει μια νέα τάση, τα υβριδικά περιβάλλοντα προγραμματισμού (Blanchard, 2017· Weintrop & Holbert, 2017· Weintrop & Wilensky, 2017) τα οποία παρέχουν τη δυνατότητα μετατροπής του κώδικα με πλακίδια στον κώδικα κάποιας γλώσσας προγραμματισμού. Ο στόχος είναι να γίνει πιο εύκολα η μετάβαση από τον οπτικό προγραμματισμό στον προγραμματισμό με κώδικα-κείμενο. Τέτοια περιβάλλοντα είναι το Alice (Cohen, 2013) και το Greenfoot (Jonas, & Sabin, 2015) που παρέχουν μια εναλλακτική αναπαράσταση του κώδικα με πλακίδια σε Java, ενώ το Pencil Code (Bau et al., 2015) και το Blockly (Fraser, 2015) χρησιμοποιούν Javascript και Python.

Όλα τα παραπάνω εργαλεία δίνουν έμφαση στην ομαλή εισαγωγή των αρχάριων σε βασικές έννοιες προγραμματισμού υπολογιστών, χρησιμοποιώντας μια εναλλακτική

παρουσίαση του κώδικα του προγράμματος. Ενώ με την προσέγγιση αυτή αποφεύγονται τα προβλήματα που αντιμετωπίζουν οι αρχάριοι προγραμματιστές κατά τη σύνταξη του προγράμματος, οι δυσκολίες που εμφανίζονται στην κατανόηση της λειτουργίας βασικών αλγορίθμων σε δομές δεδομένων παραμένουν. Είναι επιτακτική η ανάγκη για εκπαιδευτικά εργαλεία τα οποία θα παρουσιάζουν με κάποιο επεξηγηματικό τρόπο τα βασικά χαρακτηριστικά της λειτουργίας του αλγορίθμου έτσι ώστε αυτή να γίνεται κατανοητή από τους αρχάριους προγραμματιστές.

Στο πλαίσιο αυτό, έχει προταθεί ο σχεδιασμός κατάλληλων μαθησιακών δραστηριοτήτων με χρήση εκπαιδευτικών περιβαλλόντων προσομοίωσης-οπτικοποίησης αλγορίθμων και προγραμμάτων (Stasko, 1997· Hundhausen & Douglas, 2002· Sajaniemi & Kuittinen, 2003· Vrachnos & Jimoyiannis, 2008· 2009· 2014). Η σχετική βιβλιογραφία είναι αρκετά εκτεταμένη και μια αναλυτική επισκόπηση παρουσιάζεται στο 3$^{ο}$ κεφάλαιο της παρούσας διατριβής.

# 3

# *Επισκόπηση Εναλλακτικών Εκπαιδευτικών Περιβαλλόντων Προγραμματισμού*

## *3.1 Περιβάλλοντα Οπτικοποίησης*

### *3.1.1 Εισαγωγή*

Ένας σημαντικός αριθμός περιβαλλόντων οπτικοποίησης αλγορίθμων έχουν αναπτυχθεί και είναι διαθέσιμα για εκπαιδευτική χρήση στα εισαγωγικά μαθήματα προγραμματισμού, κυρίως σε πανεπιστημιακό επίπεδο (Cross & Hendrix, 2006· Hundhausen & Brown, 2007· Moreno et al., 2004). Οι εφαρμογές αυτές έχουν σημαντικές διαφορές στον τρόπο υλοποίησης τους, και γενικά χαρακτηρίζονται από μεγάλη ετερογένεια. Οι διαφορές εντοπίζονται στον τρόπο παρουσίασης, στο βαθμό αλληλεπίδρασης του μαθητή με το περιβάλλον, στον τρόπο ανάπτυξης της οπτικοποίησης και σε άλλα σημεία.

Στο κεφάλαιο αυτό παρουσιάζονται τα σημαντικότερα εργαλεία οπτικοποίησης αλγορίθμων που χρησιμοποιούνται σήμερα διεθνώς. Αναφέρονται επίσης, κάποια παλαιότερα συστήματα τα οποία είχαν σημαντική συμβολή στην ανάπτυξη του συγκεκριμένου χώρου. Για την κατάταξη των συστημάτων χρησιμοποιείται η ταξινομία του Price (1993) σε συνδυασμό με την ταξινομία των επιπέδων διαδραστικότητας (αλληλεπίδρασης) που έχει προταθεί από τους Naps et al. (2003).

Ο στόχος μας είναι αρχικά η παρουσίαση και ταξινόμηση των διαθέσιμων εκπαιδευτικών περιβαλλόντων οπτικοποίησης, έτσι ώστε να εντοπιστούν και να αναδειχθούν τα τεχνολογικά και τα παιδαγωγικά χαρακτηριστικά τους. Η καταγραφή και σύγκριση των παραπάνω συστημάτων έγινε με σκοπό τον καθορισμό των βασικών

χαρακτηριστικών και προδιαγραφών που πρέπει να έχει ένα ιδανικό σύστημα οπτικοποίησης αλγορίθμων, ώστε να είναι αποτελεσματικό στην εκπαιδευτική διαδικασία.

### *3.1.2 Θεμελιώδεις Έννοιες*

Με τον όρο οπτικοποίηση (visualization) αλγορίθμου περιγράφεται μια διαδραστική οπτικοποιημένη παρουσίαση της λογικής του αλγορίθμου, βασισμένη σε μια σειρά εικόνων και αναπαραστάσεων που αναδεικνύουν τα βασικά χαρακτηριστικά της συμπεριφοράς του (Hundhausen & Brown, 2007). Τα συνήθη λογισμικά προσομοίωσης-οπτικοποίησης δεν υποκαθιστούν το προγραμματιστικό περιβάλλον, αφού δεν εκτελούν κάποιον αλγόριθμο, αλλά παρουσιάζουν, οπτικοποιούν και προσομοιώνουν την εκτέλεσή του για προκαθορισμένα δεδομένα εισόδου.

Υπάρχουν πολλοί λόγοι που συνηγορούν στη χρήση κατάλληλα σχεδιασμένων περιβαλλόντων προσομοίωσης-οπτικοποίησης σε εισαγωγικά μαθήματα προγραμματισμού. Οι προσομοιώσεις αλγορίθμων αποτελούν δυναμικά μαθησιακά περιβάλλοντα τα οποία (Τζιμογιάννης κ.α., 2006):

- επιτρέπουν την οπτικοποίηση της λειτουργίας ενός αλγορίθμου, υποστηρίζοντας τη διάλεξη του εκπαιδευτικού και την εργαστηριακή εξάσκηση των μαθητών.
- επιτρέπουν στους μαθητές να απομονώσουν τις μεταβλητές και το ρόλο τους στο πρόγραμμα, με στόχο την κατανόηση σύνθετων υπολογιστικών δομών και διαδικασιών.
- βοηθούν τους μαθητές να εμβαθύνουν στη λογική του προγραμματισμού και να κατανοήσουν δύσκολες υπολογιστικές έννοιες και διαδικασίες.
- ενεργοποιούν το ενδιαφέρον των μαθητών, παρέχοντας δυνατότητες να εκφράσουν τις δικές τους αναπαραστάσεις για αλγόριθμους και διαδικασίες.
- διευκολύνουν την ενεργή μάθηση μέσα από διαδικασίες παράθεσης υποθέσεων, μεταβολής των τιμών εισόδου και άμεσου ελέγχου των αποτελεσμάτων στην οθόνη.
- βοηθούν τους μαθητές να μάθουν μέσα από διερευνητικές δραστηριότητες.

Οι περισσότερες έρευνες έχουν δείξει ότι τα λογισμικά προσομοίωσης αλγορίθμων θα πρέπει να προωθούν τη διερευνητική μάθηση και να ευνοούν την ενεργητική συμμετοχή και όχι την παθητική συμμόρφωση των μαθητών (Hundhausen & Brown, 2007· Hundhausen, Douglas & Stasko, 2002· Urquiza–Fuentes, & Velazquez–Iturbide, 2009). Ο μαθητής δεν αρκεί απλά να παρακολουθεί την οπτικοποίηση ως παθητικός θεατής αλλά θα πρέπει να πειραματιστεί με την εκτέλεση του αλγορίθμου, έτσι ώστε να αναδειχθούν τα ιδιαίτερα χαρακτηριστικά της λειτουργίας του. Ένα σύγχρονο εκπαιδευτικό περιβάλλον οπτικοποίησης θα πρέπει να παρέχει στο μαθητή δυνατότητες να πειραματιστεί με την εκτέλεση του αλγορίθμου, να αλλάξει παραμέτρους, να ελέγξει ιδέες και αναπαραστάσεις, έτσι ώστε να ανακαλύψει τις διάφορες πτυχές του και να οικοδομήσει επαρκείς και λειτουργικές αναπαραστάσεις με στόχο την εφαρμογή τους για την επίλυση νέων προβλημάτων.

Σε κάθε εκπαιδευτική προσομοίωση διακρίνονται τρεις βασικές συνιστώσες (Jimoyiannis, 2009): α) το σενάριο της προσομοίωσης (simulation scenario), β) το υποκείμενο μοντέλο του συστήματος (model) και γ) η διδακτική συνιστώσα (instructional overlay), που καθορίζεται από τα αναπαραστατικά εργαλεία, την παιδαγωγική φιλοσοφία (ανακαλυπτική ή διερευνητική μάθηση) και τις χρησιμοποιούμενες δραστηριότητες. Οι παραπάνω συνιστώσες ορίζουν τον τρόπο με τον οποίο προσεγγίζεται και υποστηρίζεται η νέα γνώση από το εργαλείο προσομοίωσης.

Μια εφαρμογή οπτικοποίησης στοχεύει σε μια επεξηγηματική αναπαραγωγή της εκτέλεσης ενός αλγορίθμου, ώστε ο μαθητής να ανακαλύψει σημαντικά χαρακτηριστικά της συμπεριφοράς του αλγορίθμου. Επίσης, μπορεί να αποτελέσει ένα εργαλείο επιβεβαίωσης της ορθότητας του αλγορίθμου που έχει σχεδιάσει ο μαθητής, μέσα από την δυναμική οπτικοποίησή του.

Η οπτικοποίηση γενικότερα αποτελεί ένα μέσο διευκόλυνσης του εκπαιδευτικού για την επεξήγηση της λειτουργίας διάφορων βασικών εννοιών της πληροφορικής, από τη νοητή μηχανή που περιγράφει το μοντέλο λειτουργίας του υπολογιστή μέχρι τους αλγορίθμους εύρεσης συντομότερου μονοπατιού και ελάχιστου συνδετικού δέντρου σε γράφους. Τα τελευταία χρόνια η χρήση οπτικοποιήσεων για τη διδασκαλία του προγραμματισμού κερδίζει συνεχώς έδαφος μεταξύ των εκπαιδευτικών (Naps et. al. 2003). Σύμφωνα με τις βασικές αρχές του εποικοδομισμού οι οπτικοποιήσεις θα πρέπει να υιοθετούν ένα μοντέλο που να είναι οικείο στους μαθητές και το οποίο να

επικεντρώνει στις ιδιότητες των προγραμματιστικών εννοιών που θέλουμε να εξηγήσουμε. Ένα τέτοιο μοντέλο αποτελεί η οπτικοποίηση μιας μεταβλητής ως ένα κουτί ή γραμματοκιβώτιο το περιεχόμενο του οποίου μπορεί να μεταβάλλεται. Μια άλλη περίπτωση τέτοιου μοντέλου αποτελεί η αναλογία μεταξύ μιας κλάσης και ενός συρταριού που περιέχει φακέλους, οι οποίοι παίζουν τον ρόλο των αντικειμένων (Gries, 2008).

Η αξιοποίηση των οπτικοποιήσεων αλγορίθμων για εκπαιδευτικούς σκοπούς έχει μακρά ιστορία στην εκπαίδευση. Το βίντεο *Sorting Out Sorting* (Baecker, 1981), το οποίο παρουσίαζε εικόνες δεδομένων που ταξινομούνταν από διαφορετικούς αλγορίθμους θεωρείται η πρώτη σημαντική αναφορά στην οπτικοποίηση αλγορίθμων. Από τότε έχουν αναπτυχθεί πολλά συστήματα προσομοίωσης και οπτικοποίησης αλγορίθμων που βασίζονταν στις διαθέσιμες κάθε φορά τεχνολογίες.

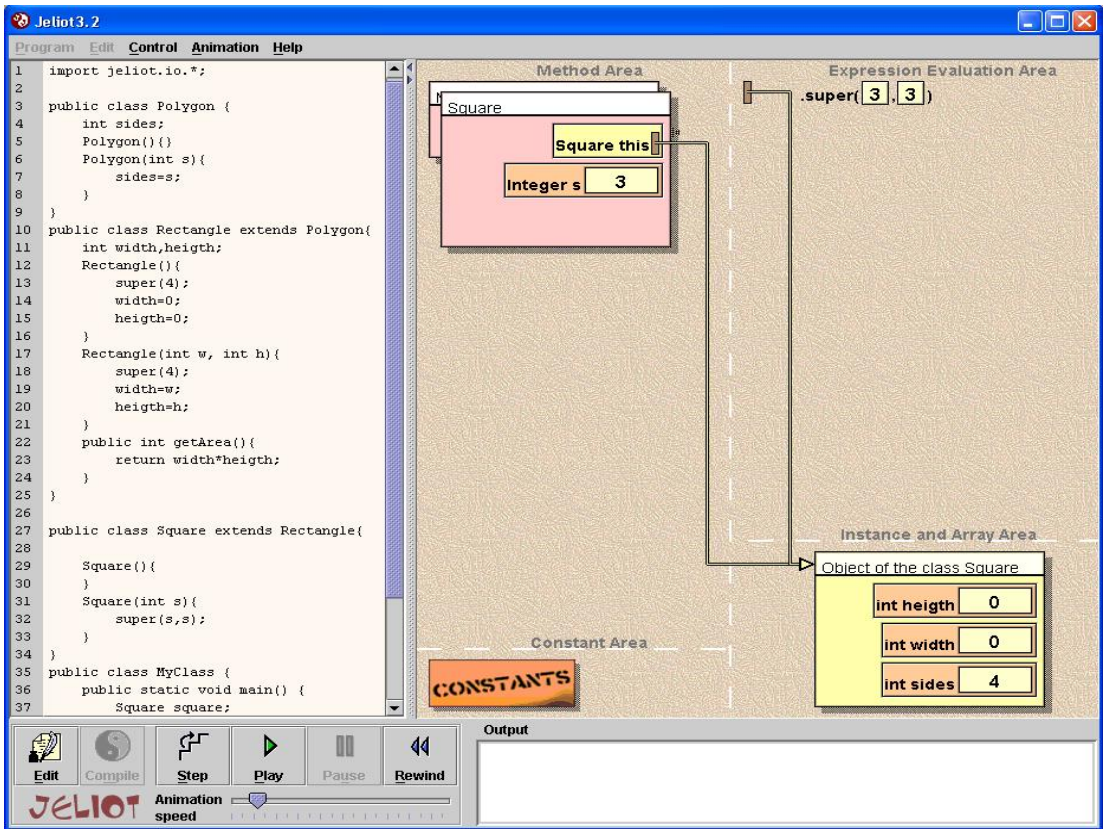
Ένας σημαντικός αριθμός περιβαλλόντων οπτικοποίησης έχουν αναπτυχθεί και είναι διαθέσιμα για εκπαιδευτική χρήση στα εισαγωγικά μαθήματα προγραμματισμού, κυρίως σε πανεπιστημιακό επίπεδο (Cross & Hendrix, 2006· Hundhausen & Brown, 2007· Moreno et al., 2004). Οι εφαρμογές αυτές έχουν σημαντικές διαφορές στον τρόπο υλοποίησης τους, και γενικά χαρακτηρίζονται από μεγάλη ετερογένεια. Οι διαφορές εντοπίζονται στον τρόπο παρουσίασης, στο βαθμό αλληλεπίδρασης του μαθητή με το περιβάλλον, στον τρόπο ανάπτυξης της οπτικοποίησης και σε άλλα σημεία. Στη συνέχεια παρουσιάζονται τα σημαντικότερα εκπαιδευτικά περιβάλλοντα οπτικοποίησης στον προγραμματισμό, τα οποία διακρίνονται σε δύο κατηγορίες με βάση τα χαρακτηριστικά και τις δομές του αλγορίθμου που οπτικοποιούν: α) συστήματα οπτικοποίησης αλγορίθμων και β) συστήματα οπτικοποίησης προγραμμάτων.

## *3.2 Συστήματα Οπτικοποίησης Προγραμμάτων*

Τα συστήματα οπτικοποίησης προγραμμάτων αναπαριστούν γραφικά βασικές προγραμματιστικές δομές, όπως οι μεταβλητές, οι δείκτες ή οι πίνακες που είναι χρήσιμες στους προγραμματιστές, ειδικά κατά την εκσφαλμάτωση (debugging) των προγραμμάτων. Μέσω της οπτικοποίησης γίνεται παρακολούθηση των τιμών των μεταβλητών σε κάθε βήμα του αλγορίθμου, με στόχο την ανίχνευση και τη διόρθωση των λαθών του προγράμματος.

### *3.2.1 Το περιβάλλον αντικειμενοστρεφούς προγραμματισμού Jeliot*

Χαρακτηριστικός εκπρόσωπος της κατηγορίας αυτής είναι το σύστημα Jeliot που έχει ως στόχο την οπτικοποίηση προγραμμάτων που είναι γραμμένα στην γλώσσα προγραμματισμού Java. Ουσιαστικά πρόκειται για μια οικογένεια εφαρμογών, πρώτο μέλος της οποίας ήταν το Jeliot 2000 και τελευταίο μέλος το Jeliot 3 (Moreno et al., 2004). Η δυνατότητα που χαρακτηρίζει το Jeliot είναι η αυτόματη οπτικοποίηση των προγραμμάτων που είναι γραμμένα στη γλώσσα προγραμματισμού Java. Στο Σχήμα 3.1 απεικονίζεται μια οθόνη του περιβάλλοντος. Στο αριστερό τμήμα εμφανίζεται η τρέχουσα κατάσταση του προγράμματος (μεταβλητές, αντικείμενα και μέθοδοι) και στο δεξί η δυναμική οπτικοποίηση του υπολογισμού σύνθετων εκφράσεων και των δομών επανάληψης και επιλογής. Κάτω αριστερά βρίσκονται τα πλήκτρα ελέγχου της εκτέλεσης του προγράμματος. Η εφαρμογή εστιάζει στην οπτικοποίηση των βασικών αντικειμενοστρεφών δομών (κλάσεις, αντικείμενα, μέθοδοι κ.λπ.). Η τελευταία έκδοση του Jeliot μπορεί να προσαρτηθεί στο εκπαιδευτικό περιβάλλον BlueJ για Java (Kölling et al., 2003). Σε αυτή την περίπτωση κάποιος μπορεί να χρησιμοποιεί το περιβάλλον BlueJ και να βλέπει την οπτικοποίηση του προγράμματός του αυτόματα μέσω του Jeliot.

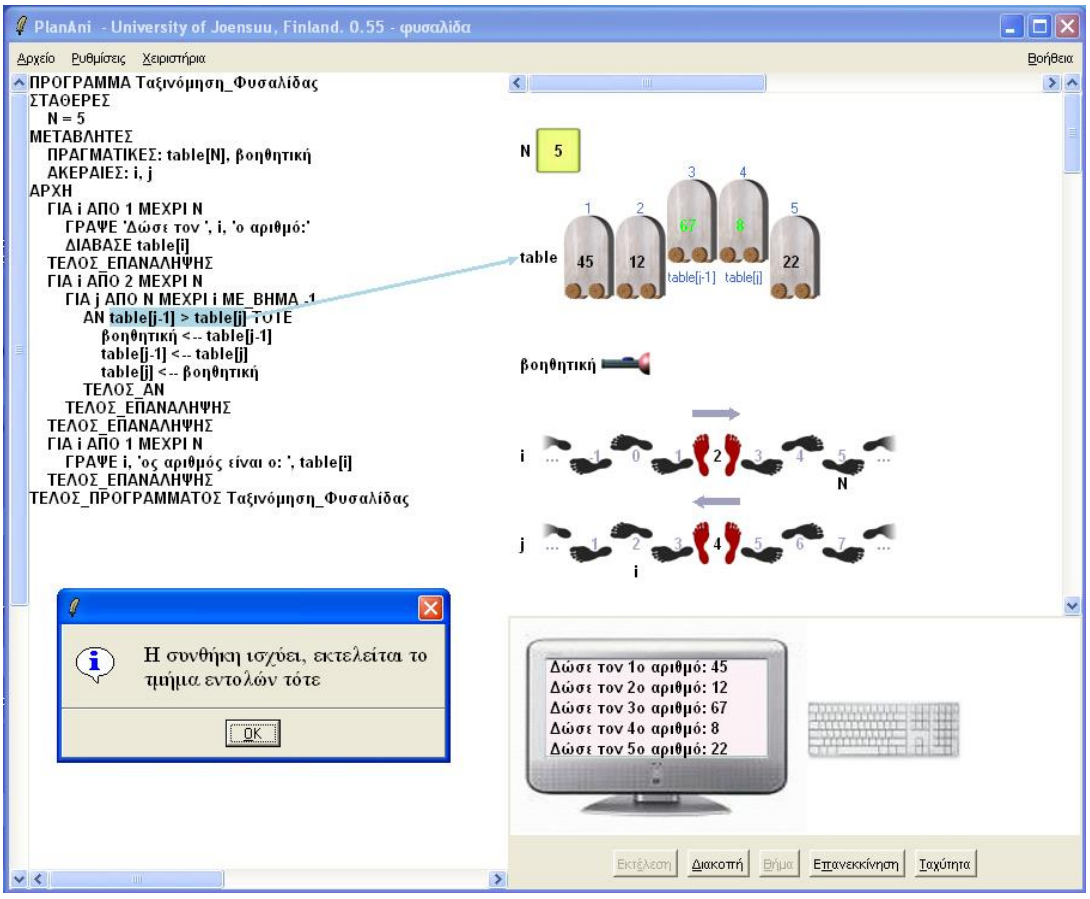



**Σχήμα 3.1 Μέθοδοι και αντικείμενα στο Jeliot**

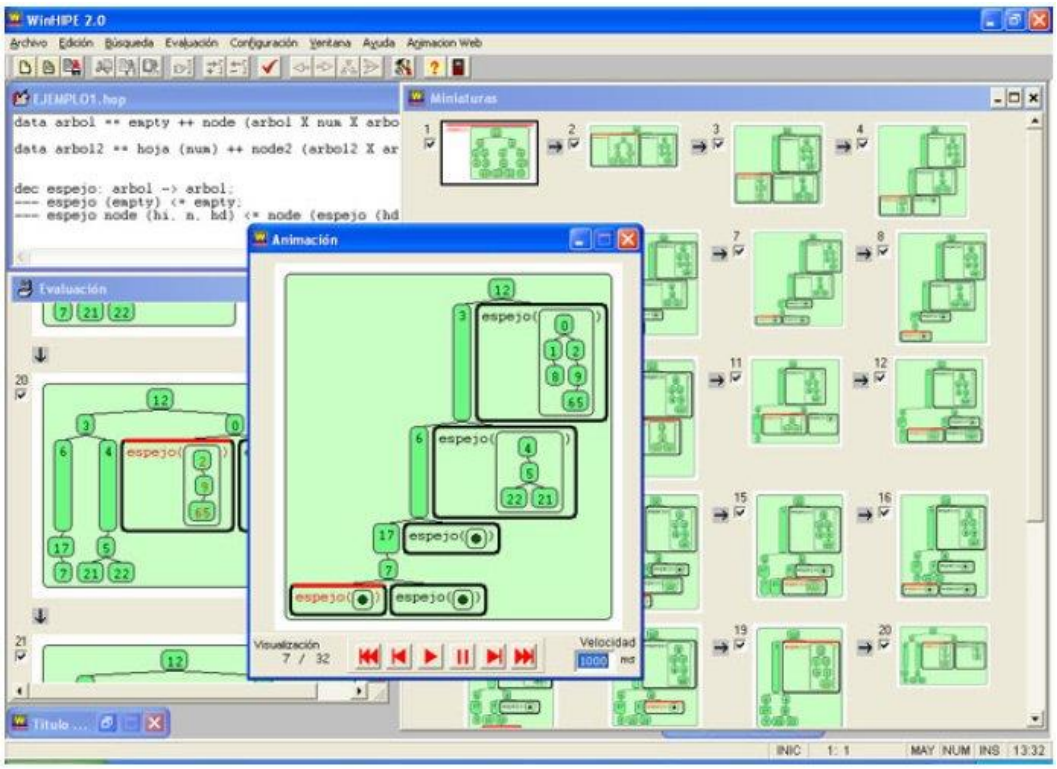
Από μια επισκόπηση των ερευνών που έγιναν σχετικά με την εκπαιδευτική αξία του (Ben-Bassat Levy & Ben-Ari, 2009), προέκυψε ότι το Jeliot α) βελτιώνει τη μάθηση στον προγραμματισμό παρέχοντας στέρεες αναπαραστάσεις των δυναμικών πτυχών ενός προγράμματος, β) εστιάζει την προσοχή των μαθητών στις βασικές αντικειμενοστρεφείς δομές και γ) επηρεάζει το επίπεδο εμπλοκής των μαθητών με το λογισμικό, ειδικά αν αυτό χρησιμοποιηθεί σε ένα συνεργατικό πλαίσιο εφαρμογής.

### *3.2.2 Το λογισμικό PlanAni*

Το PlanAni αποτελεί ένα περιβάλλον προσομοίωσης της εκτέλεσης προγραμμάτων, το οποίο βασίζεται στην προσέγγιση της διδασκαλίας με ρόλους μεταβλητών (Sajaniemi & Kuittinen, 2003). Το λογισμικό αυτό χρησιμοποιεί διαφορετική αναπαράσταση για κάθε μεταβλητή ανάλογα με το ρόλο της στο πρόγραμμα. Ο μαθητής έχει τη δυνατότητα να παρακολουθεί την προσομοίωση της εκτέλεσης του προγράμματος ορίζοντας τα δεδομένα εισόδου αλλά είναι δύσκολο να κατασκευάσει την οπτικοποίηση του δικού του αλγορίθμου, γιατί θα πρέπει να προσθέσει τις κατάλληλες εντολές οπτικοποίησης ρόλων.




**Σχήμα 3.2. Ταξινόμηση φυσαλίδας στην ελληνική έκδοση του PlanAni**




**Σχήμα 3.3. Οπτικοποίηση της αναδρομής με το WinHIPE**

Στο Σχήμα 3.2 δίνεται μια οθόνη της ελληνικής έκδοσης του PlanAni με την οπτικοποίηση του αλγόριθμου ταξινόμησης της φυσαλίδας σε ψευδογλώσσα (Τζιμογιάννης κ.α., 2006).

### *3.2.3 Τα περιβάλλοντα WinHIPE και SRec*

Το WinHIPE (Pareja-Flores et.al., 2007) δέχεται προγράμματα γραμμένα στη συναρτησιακή γλώσσα προγραμματισμού Hope, τα οποία οπτικοποιεί με σκοπό να αναδείξει τα ιδιαίτερα χαρακτηριστικά του συναρτησιακού προγραμματισμού, όπως είναι το μοντέλο αποτίμησης, οι λίστες και η αναδρομή. Το γραφικό περιβάλλον (διεπαφή) επιτρέπει τη σταδιακή εκτέλεση, την παύση ή την αντίστροφη εκτέλεση του προγράμματος, μέσω ενός οπτικού (VCR-like) τηλεχειριστηρίου (Σχήμα 3.3). Η μόνη δυνατότητα που λείπει είναι η εκτέλεση της οπτικοποίησης βήμα-βήμα, δίπλα στον κώδικα του μαθητή, με παράλληλη επισήμανση της εντολής που εκτελείται.

Η ίδια ερευνητική ομάδα έχει αναπτύξει και το περιβάλλον οπτικοποίησης SRec (Velasquez-Iturbide, Perez-Carrasco & Urquiza-Fuentes, 2008). Το σύστημα αυτό οπτικοποιεί την εκτέλεση αναδρομικών αλγορίθμων που έχουν σχεδιαστεί με την τεχνική Διαίρει και Βασίλευε και είναι διατυπωμένοι στη γλώσσα προγραμματισμού Java. Η οπτικοποίηση του αλγορίθμου παράγεται πολύ εύκολα από το χρήστη, ο οποίος

πρέπει απλά να καθορίσει την Διαίρει και Βασίλευε αναδρομική δομή του αλγορίθμου. Δηλαδή η οπτικοποίηση δεν είναι πλήρως αυτοματοποιημένη και χρειάζεται κάποια παραμετροποίηση από την πλευρά του χρήστη, ώστε να γίνει σωστά η γραφική παρουσίαση του δέντρου των αναδρομικών κλήσεων.

### *3.2.4 Το περιβάλλον JIVE*

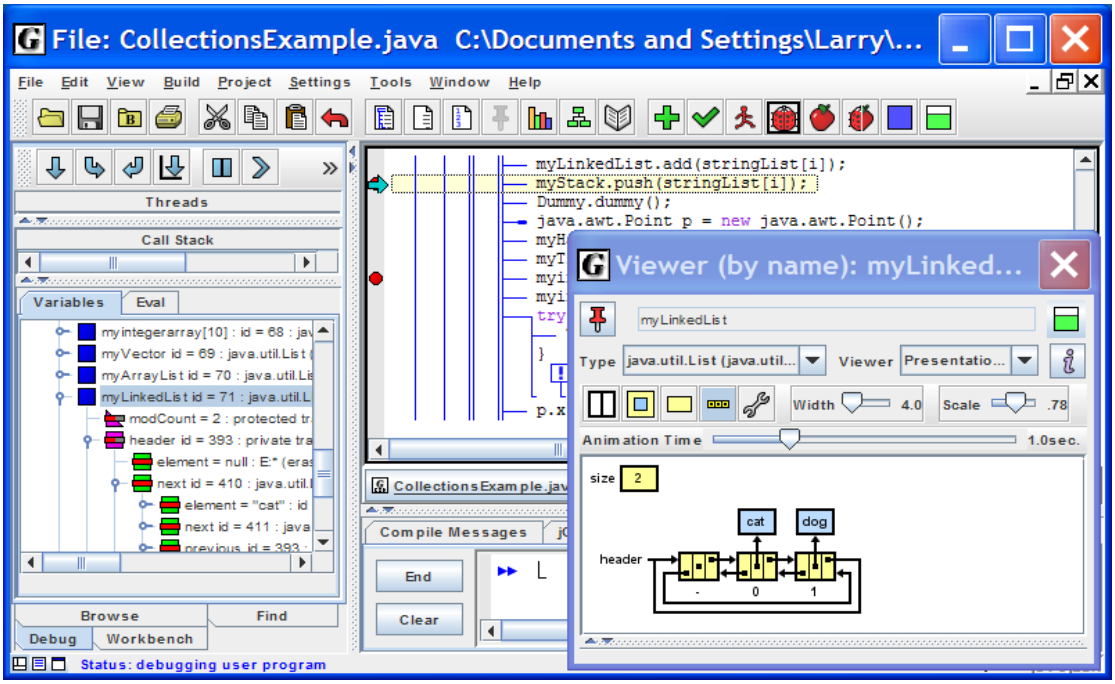
Το JIVE (Java Interactive Visualization Environment) είναι ένα περιβάλλον οπτικοποίησης αντικειμενοστρεφών δομών δεδομένων για προγράμματα που είναι γραμμένα σε Java (Gestwicki & Jayaraman, 2002). Περιλαμβάνει τα εξής χαρακτηριστικά:

- Μια γλώσσα για την περιγραφή των καταστάσεων που δημιουργούνται κατά την εκτέλεση ενός προγράμματος σε Java.
- Υψηλό βαθμό αλληλεπίδρασης με τον χρήστη κατά την εκτέλεση.
- Αντίστροφη εκτέλεση (μετάβαση σε προηγούμενη κατάσταση).
- Αυτόματη κατασκευή διαγραμμάτων αντικειμένων και ακολουθίας (object/ sequence diagrams), αντίστοιχα της γλώσσας UML.

Ένα σημαντικό χαρακτηριστικό είναι η δυνατότητα που έχει ο χρήστης να δει το χρονογράφημα όλης της εκτέλεσης (execution history), μέσα από τα διαγράμματα ακολουθίας της UML, όπου φαίνονται όλες οι καταστάσεις της εκτέλεσης του προγράμματος. Ο χρήστης μπορεί να επιλέξει μια από αυτές και να δει την οπτικοποίηση του διαγράμματος των αντικειμένων (UML).

### *3.2.5 Το περιβάλλον προγραμματισμού jGrasp*

Το jGrasp είναι ένα εκπαιδευτικό περιβάλλον προγραμματισμού με σκοπό την εκμάθηση βασικών εννοιών του αντικειμενοστρεφούς προγραμματισμού, όπως είναι η κλάση, το αντικείμενο, η κληρονομικότητα και ο πολυμορφισμός (Cross & Hendrix, 2006). Παρέχει τη δυνατότητα της αυτόματης οπτικοποίησης των περισσότερων δομών δεδομένων της Java, όπως είναι οι λίστες, οι πίνακες και οι συλλογές (collections), με στόχο την καλύτερη εκσφαλμάτωση των προγραμμάτων. Δεν πρόκειται για δυναμική οπτικοποίηση με κίνηση των αντικειμένων, αλλά για ενημέρωση των περιεχομένων των αντίστοιχων δομών. Ωστόσο οι γραφικές αναπαραστάσεις των δομών δεδομένων συγκλίνουν στη νοερή δομή που έχουμε για αυτές, όπως είναι για παράδειγμα η δομή της λίστας (Σχήμα 3.4).

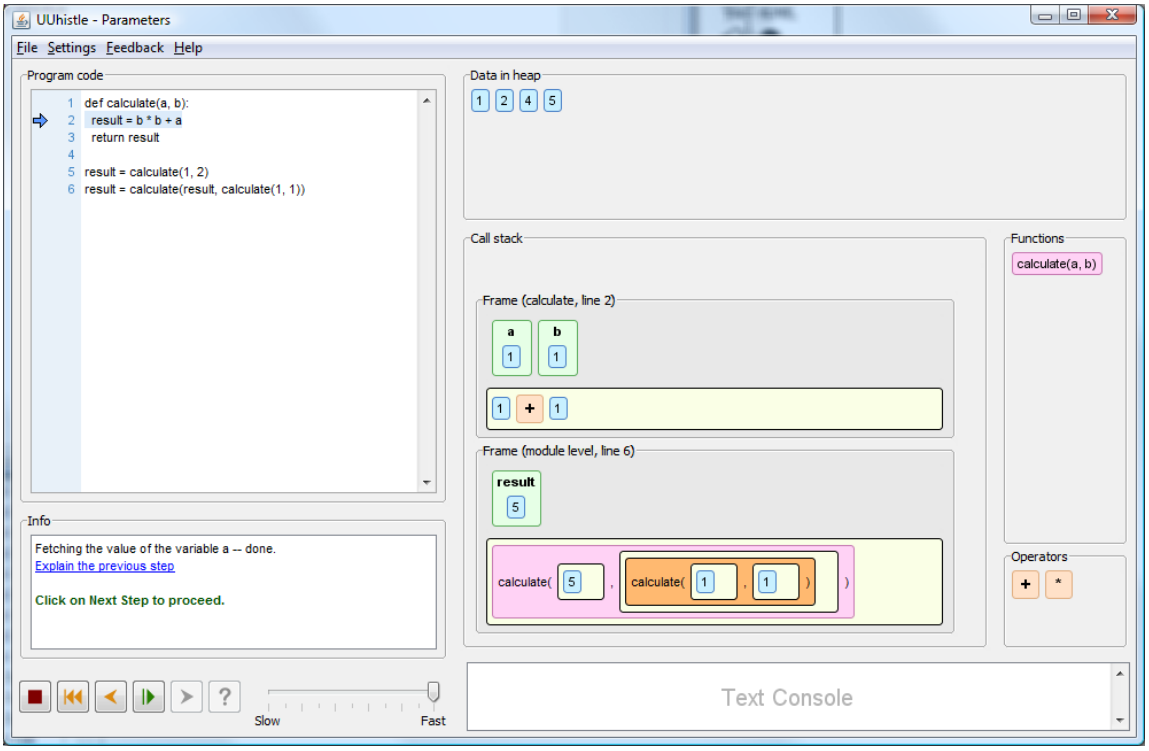



**Σχήμα 3.4. Οπτικοποίηση μιας διπλά συνδεδεμένης λίστας σε Java μέσω jGrasp**

Το λογισμικό χρησιμοποιείται σε περισσότερα από 300 εκπαιδευτικά ιδρύματα για τη διδασκαλία της γλώσσας προγραμματισμού Java. Υποστηρίζει πολλαπλές όψεις του ίδιου τμήματος κώδικα, όπως είναι το UML διάγραμμα και ο πάγκος εργασίας (workbench) για τα αντικείμενα που δημιουργούνται κατά την εκτέλεση. Έχει τη δυνατότητα εκτέλεσης μεμονωμένων εντολών με ταυτόχρονη ενημέρωση όλων των όψεων που έχει ανοίξει ο χρήστης, όπως ο πάγκος εργασίας (workbench) και η οπτικοποίηση της δομής στη μνήμη (viewer).

### *3.2.6 Το περιβάλλον UUhistle*

Το UUhistle είναι ένα περιβάλλον οπτικοποίησης προγραμμάτων στη γλώσσα Python (Sorva & Sirkia, 2010). Υποστηρίζει διάφορα επίπεδα διαδραστικότητας του χρήστη με την οπτικοποίηση του προγράμματος. Ο χρήστης μπορεί να ελέγχει την ταχύτητα της οπτικοποίησης. Το περιβάλλον λειτουργεί ως διερμηνευτής όπως στα πρότυπα της γλώσσας Python, οπτικοποιώντας την εκτέλεση των εντολών με τη σειρά που δίνονται χωρίς να περιμένει να συμπληρωθεί ολόκληρο το πρόγραμμα. Το UUhistle υποστηρίζει τη δημιουργία ασκήσεων οπτικής προσομοίωσης προγραμμάτων python από τον καθηγητή, στις οποίες μπορούν να ενσωματωθούν ερωτήσεις προς τον μαθητή σε σημαντικά σημεία της εκτέλεσης του προγράμματος. Οι ερωτήσεις εμφανίζονται στον μαθητή με τη μορφή αναδυόμενων παραθύρων.

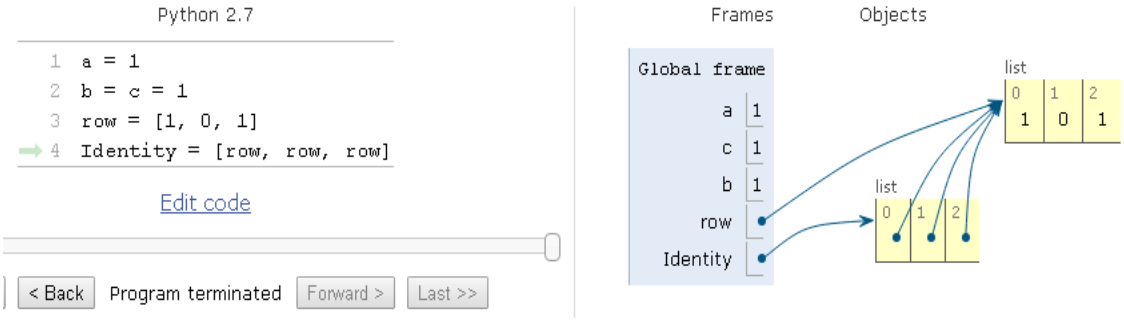



**Σχήμα 3.5. Οπτικοποίηση ενός απλού υπολογισμού με το UUhistle**

### *3.2.7 Online Python Tutor*

Το Online Python Tutor (Guo, 2012) είναι ένα περιβάλλον οπτικοποίησης προγραμμάτων στη γλώσσα προγραμματισμού Python το οποίο διατίθεται δωρεάν και είναι ανοικτού κώδικα. Ένα βασικό πλεονέκτημα που έχει σε σχέση με το UUHistle είναι ότι δεν χρειάζεται εγκατάσταση αλλά εκτελείται μέσα από έναν browser, δηλαδή είναι ένα διαδικτυακό περιβάλλον το οποίο δεν χρειάζεται την εγκατάσταση κάποιου πρόσθετου λογισμικού. Έτσι μπορεί να χρησιμοποιηθεί και από άλλες συσκευές όπως tablets και από οποιονδήποτε με πρόσβαση στο διαδίκτυο.

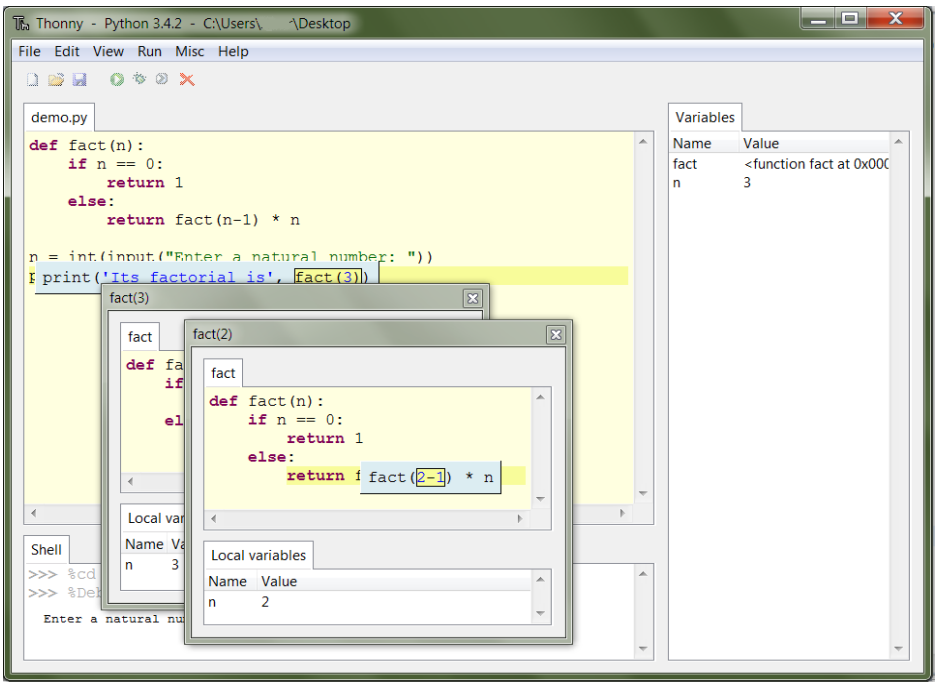
Με το python tutor οι μαθητές μπορούν να δώσουν εντολές σε Python και να παρατηρήσουν την οπτικοποίηση εσωτερικών δομών της Python, όπως είναι οι λίστες, οι πλειάδες, τα σύνολα και τα λεξικά. Στο Σχήμα 3.6 φαίνεται η οπτικοποίηση ενός πίνακα δύο διαστάσεων που αναπαρίσταται στην Python ως λίστα από λίστες. Το Python tutor χρησιμοποιείται σε πολλά μαθήματα διεθνών πανεπιστημίων (UC Berkeley, MIT). Ένα ακόμα πλεονέκτημά του είναι ότι μπορεί να ενσωματωθεί σε μια άλλη ιστοσελίδα. Έτσι ο εκπαιδευτικός μπορεί να ενσωματώσει το εργαλείο στην ιστοσελίδα του μαθήματός του ορίζοντας μάλιστα και ένα συγκεκριμένο παράδειγμα για εκκίνηση της μαθησιακής διαδικασίας.




**Σχήμα 3.6. Οπτικοποίηση μεταβλητών και λιστών της Python στο python tutor**

Όπως φαίνεται στο Σχήμα 3.6 η οπτικοποίηση των μεταβλητών γίνεται με το κλασικό μοντέλο ως κουτί με περιεχόμενο την τιμή και όχι ως μια ετικέτα η οποία είναι συνδεδεμένη με κάποιο τρόπο με την τιμή η οποία είναι ένα ξεχωριστό αντικείμενο. Το Python tutor ξεκίνησε με αποκλειστικό σκοπό την οπτικοποίηση προγραμμάτων σε Python αλλά τα τελευταία χρόνια έχει επεκταθεί ώστε να οπτικοποιεί τις δομές και άλλων γλωσσών όπως Java, Ruby, JavaScript, Racket και άλλες.

Εκτός από το UUHistle και το Python tutor πρόσφατα έχουν αναπτυχθεί διάφορα περιβάλλοντα οπτικοποίησης για προγράμματα γραμμένα στη γλώσσα Python όπως για παράδειγμα το Thonny (Annamaa, 2015) το οποίο παρουσιάζει γραφικά όχι μόνο την κατάσταση των μεταβλητών και των δομών δεδομένων στη μνήμη, αλλά και εμφωλευμένες κλήσεις συναρτήσεων (Σχήμα 3.7).




**Σχήμα 3.7. Αναδρομικός υπολογισμός του παραγοντικού στο Thonny**

## *3.3 Συστήματα Οπτικοποίησης Αλγορίθμων*

Τα συστήματα οπτικοποίησης αλγορίθμων είναι περιβάλλοντα που παράγουν μια γραφική παρουσίαση της εκτέλεσης του αλγορίθμου με σκοπό να αναδειχθούν τα ιδιαίτερα χαρακτηριστικά του, τα οποία είναι δύσκολο να εντοπίσουν οι μαθητές (Hundhausen & Brown, 2007· Rössling & Freisleben, 2002· Vrachnos & Jimoyiannis, 2008· 2014).

Η πρώτη σημαντική αναφορά οπτικοποίησης αλγορίθμων είναι το γνωστό πλέον βίντεο Sorting Out Sorting (Baecker, 1981), το οποίο θεωρείται η πρώτη σημαντική αναφορά στην οπτικοποίηση αλγορίθμων. Παρουσιάζει την οπτικοποίηση διάφορων αλγορίθμων ταξινόμησης πινάκων με προκαθορισμένα δεδομένα ώστε να κατανοήσουν οι σπουδαστές τη λειτουργία τους. Από τότε έχουν αναπτυχθεί πολλά συστήματα οπτικοποίησης αλγορίθμων με σημαντικές εφαρμογές στην εκπαιδευτική πρακτική (Brown, 1991· Hundhausen & Brown, 2007· Rössling & Freisleben, 2002). Στην παράγραφο αυτή θα παρουσιαστούν τα πιο ενδιαφέροντα (σε τεχνολογικό και παιδαγωγικό επίπεδο) και δημοφιλή συστήματα οπτικοποίησης αλγορίθμων.

### *3.3.1 Το σύστημα Zeus*

Το σύστημα Zeus (Brown, 1991) προέρχεται από την οικογένεια συστημάτων BALSA και BALSA II. Το σύστημα BALSA (Brown Algorithm Simulator and Animator) θεωρείται ως ο πρόγονος των σύγχρονων εφαρμογών οπτικοποίησης αλγορίθμων. Με το σύστημα αυτό ο Brown εισήγαγε τη έννοια του ενδιαφέροντος γεγονότος (interesting event) κατά την εκτέλεση ενός αλγορίθμου. Ένα ενδιαφέρον γεγονός μπορεί να είναι η αλλαγή της τιμής μιας μεταβλητής ή η αφαίρεση ενός αντικειμένου από τη στοίβα. Με κατάλληλη επιλογή των ενδιαφερόντων γεγονότων ενός αλγορίθμου, επιλέγουμε ουσιαστικά τις πτυχές του αλγορίθμου που έχουν διδακτικό-μαθησιακό ενδιαφέρον ή δυσκολία και πρέπει να οπτικοποιηθούν. Έτσι μπορούμε να παρακολουθούμε πολλαπλές όψεις της εκτέλεσης του αλγορίθμου να εξελίσσονται ταυτόχρονα.

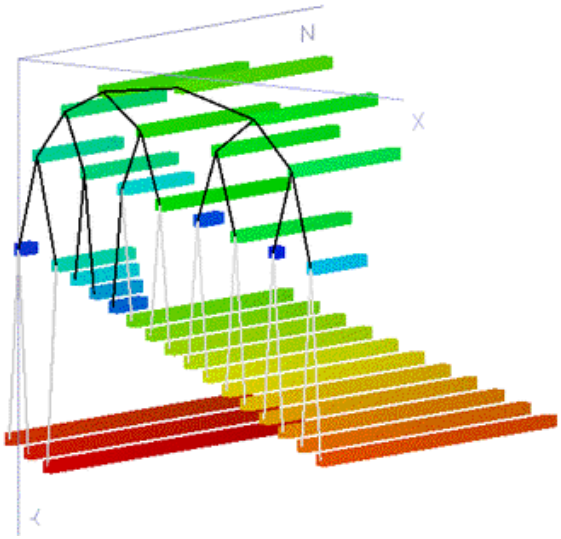
Για παράδειγμα μια δημοφιλής όψη ενός αλγορίθμου ταξινόμησης, παρουσιάζει τα στοιχεία του πίνακα προς ταξινόμηση ως ένα σύνολο ιστογραμμάτων (Σχήμα 3.8). Το ύψος κάθε ιστογράμματος είναι ανάλογο με την τιμή του στοιχείου του πίνακα που παριστάνει. Το Zeus είναι το πιο σύγχρονο σύστημα αυτής της κατηγορίας το οποίο μάλιστα δίνει τη δυνατότητα χρήσης τρισδιάστατων γραφικών. Το μειονέκτημα αυτής

της οικογένειας συστημάτων είναι ότι η πολυπλοκότητα της κατασκευής μια νέας οπτικοποίησης, καθιστά αυτή τη δυνατότητα αποτρεπτική όχι μόνο για τους μαθητές αλλά και για τους εκπαιδευτικούς.

### *3.3.2 Η οικογένεια συστημάτων οπτικοποίησης Tango, Polka, Samba*

Μια πολύ σημαντική οικογένεια εφαρμογών για το χώρο των συστημάτων οπτικοποίησης αλγορίθμων ξεκίνησε με τη δημιουργία του συστήματος Tango (Stasko, 1990). Το χαρακτηριστικό αυτού του συστήματος είναι η τεχνική της πορείας μετάβασης (path transition) μεταξύ των καταστάσεων. Εκεί οφείλεται και το όνομα του συστήματος (**T**ransition–based **A**nimation **G**eneration). Δηλαδή η μετάβαση των προγραμματιστικών αντικειμένων (μεταβλητές) σε μια άλλη κατάσταση παρουσιάζεται στην οθόνη με τη βαθμιαία κίνηση των αντικειμένων.

Ο αλγόριθμος που θα οπτικοποιηθεί πρέπει να είναι γραμμένος στη γλώσσα C. Ο χρήστης καθορίζει τα ενδιαφέροντα γεγονότα που θέλει να οπτικοποιήσει, εισάγοντας εντολές οπτικοποίησης στα κατάλληλα σημεία του αλγορίθμου. Μετά το Tango ακολούθησε το XTango μια έκδοση με φιλική γραφική διεπαφή για το περιβάλλον των X-Windows στο Unix. Ωστόσο, οι δυναμικές παρουσιάσεις που μπορούσαν να δημιουργηθούν με τα εργαλεία αυτά ήταν σχετικά απλές, χωρίς πολλαπλές μεταβάσεις. Δεν υποστήριζαν δηλαδή την εκτέλεση παράλληλων διεργασιών. Για το σκοπό αυτό αναπτύχθηκε λίγο αργότερα το σύστημα POLKA (**P**arallel program–focused **O**bject–oriented **L**ow **K**ey **A**nimation) (Stasko & Kraemer, 1993).




**Σχήμα 3.8. Αναπαράσταση της ταξινόμησης HeapSort στο σύστημα Zeus**

Ένα από τα τελευταία συστήματα που ανέπτυξε η ομάδα του Stasko, είναι το Samba. Πρόκειται ουσιαστικά για μια γραφική διεπαφή (front-end) και ένα διερμηνευτή, ο οποίος τροφοδοτεί με εντολές οπτικοποίησης το σύστημα Polka. Για να δημιουργήσουν οι μαθητές τις δικές τους οπτικοποιήσεις, θα πρέπει να μάθουν να χρησιμοποιούν την ειδική γλώσσα του Samba, κάτι το οποίο είναι πολύ δύσκολο για όσους παρακολουθούν ένα εισαγωγικό μάθημα αλγορίθμων.

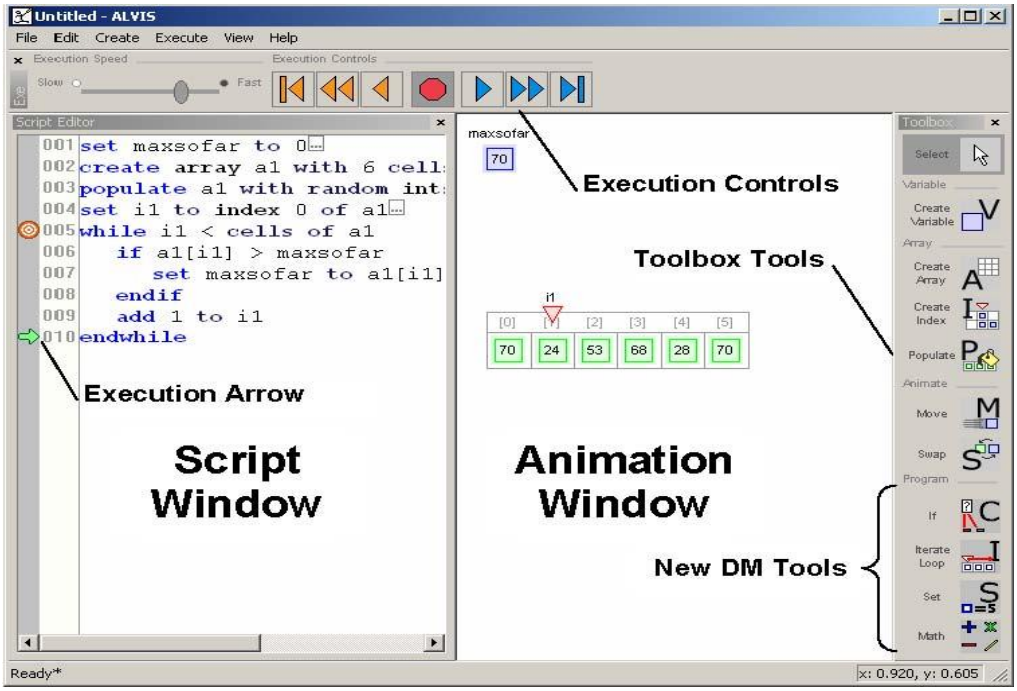
Επειδή το Samba δεν μπορεί να λειτουργήσει σε όλες τις πλατφόρμες, αναπτύχθηκε μια νέα έκδοσή του γραμμένη σε Java, το JSamba, το οποίο είναι προσπελάσιμο από το Διαδίκτυο (JSamba, 1998). Ωστόσο δε μπορεί ο χρήστης να ορίσει τη δική του οπτικοποίηση, αλλά να δει μόνο τα έτοιμα παραδείγματα που προσφέρονται. Λειτουργεί δηλαδή, σαν ένα περιβάλλον παρουσίασης οπτικοποιήσεων που είναι ήδη γραμμένες στη γλώσσα σεναρίων του Samba.

### *3.3.3 Alvis*

Η εφαρμογή Alvis Live! (**Al**gorithm **Vi**sualization **S**toryboarder) (Hundhausen & Brown, 2007) είναι ένα διαδραστικό σύστημα οπτικοποίησης αλγορίθμων, το οποίο αποτελεί μετεξέλιξη του συστήματος οπτικοποίησης Alvis (Hundhausen & Douglas, 2002). Το Alvis εκτελεί έναν αλγόριθμο για προκαθορισμένες εισόδους ή για μικρό πλήθος δεδομένων, χωρίς να χρησιμοποιεί υψηλής ποιότητας γραφικά.

Ενώ το Alvis χρησιμοποιούσε αρχικά τη γλώσσα συγγραφής σεναρίων SALSA (Spatial Algorithmic Language for StoryboArding) για την περιγραφή του αλγορίθμου και της οπτικοποίησής του, το Alvis Live! υποστηρίζει στην τελευταία έκδοσή του τη γλώσσα Cb. Πρόκειται για ένα υποσύνολο της γλώσσας C εφοδιασμένο με κάποια στοιχεία της γλώσσας σεναρίων SALSA που χρησιμοποιούσε η προηγούμενη έκδοση. Στο Σχήμα 3.9 φαίνεται μια οθόνη του Alvis. Ο μαθητής θα πρέπει να κωδικοποιήσει τον αλγόριθμό του στη γλώσσα Cb (script window). Το σύστημα υποστηρίζει τη δυναμική οπτικοποίηση του αλγορίθμου (animation window) με παράλληλη επισήμανση της τρέχουσας εντολής που εκτελείται.

Ένα άλλο χαρακτηριστικό του Alvis είναι ο υψηλός βαθμός αλληλεπίδρασης με το χρήστη, η δυνατότητα της αντίστροφης εκτέλεσης του αλγορίθμου σε προηγούμενη κατάσταση και η σταδιακή εκτέλεση του αλγορίθμου (βήμα προς βήμα).



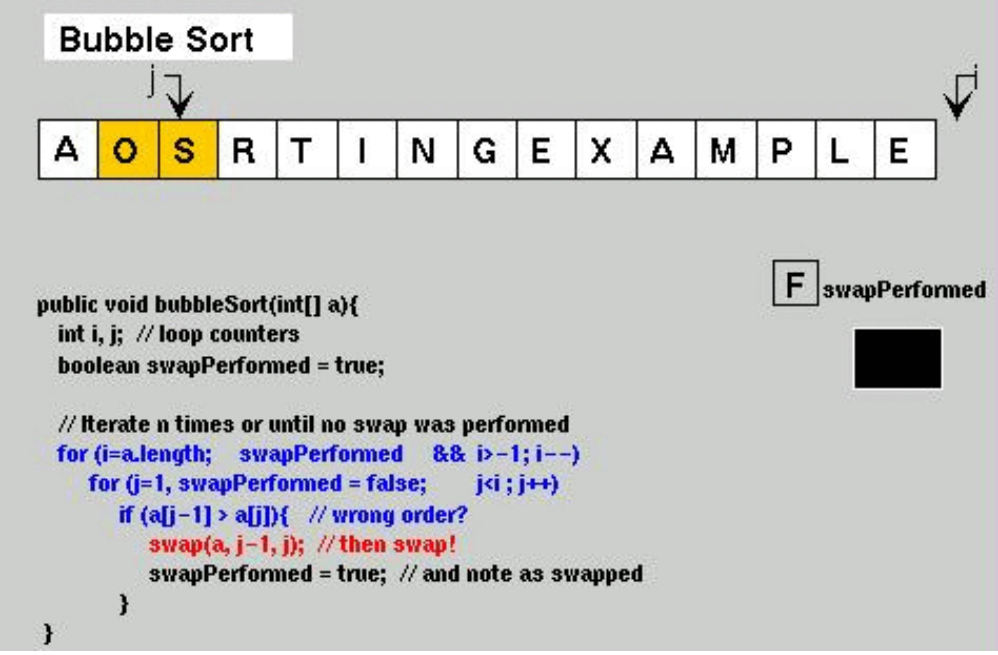

**Σχήμα 3.9. Το περιβάλλον Alvis Live!**

### *3.3.4 Animal*

Το Animal είναι ένα ισχυρό εργαλείο δημιουργίας οπτικοποιήσεων αλγορίθμων (Rössling & Freisleben, 2002). Περιλαμβάνει έναν γραφικό συντάκτη οπτικοποιήσεων, μια γλώσσα συγγραφής σεναρίων οπτικοποίησης και μια βιβλιοθήκη γραφικών (Σχήμα 3.10). Ο γραφικός συντάκτης χρησιμοποιείται όπως σε μια γλώσσα οπτικού προγραμματισμού. Ο χρήστης καλείται να σχεδιάσει οπτικά το περιβάλλον και να καθορίσει τα βασικά χαρακτηριστικά της οπτικοποίησης, Σε κάποιες περιπτώσεις, μπορεί να ορίσει πλήρως την οπτικοποίηση με γραφικό τρόπο. Στις περισσότερες όμως θα χρειαστεί να γράψει κώδικα, ώστε να συνδέσει τον αλγόριθμο με την οπτικοποίηση. Το Animal προσφέρει δύο τρόπους για να γίνει η οπτικοποίηση:

α) με τη χρήση μιας γλώσσας σεναρίων που ονομάζεται AnimalScript. Η γλώσσα αυτή υποστηρίζει και δομές δεδομένων, όπως οι πίνακες και οι λίστες.

β) με τη χρήση της βιβλιοθήκης που παρέχει το εργαλείο. Πρόκειται για μια βιβλιοθήκη σε Java (Java API). Αφού υλοποιήσει κάποιος τον αλγόριθμο (σε Java), στη συνέχεια προσθέτει στον κώδικα του αλγορίθμου τις κατάλληλες κλήσεις συναρτήσεων της βιβλιοθήκης, που έχουν ως αποτέλεσμα την παραγωγή κώδικα στη γλώσσα AnimalScript, ο οποίος στη συνέχεια θα εκτελεστεί από το Animal. Έχει αναπτυχθεί και ένα πρόσθετο (Animalipse plugin) της γλώσσας AnimalScript για το περιβάλλον Eclipse (Rössling & Schroeder, 2009).

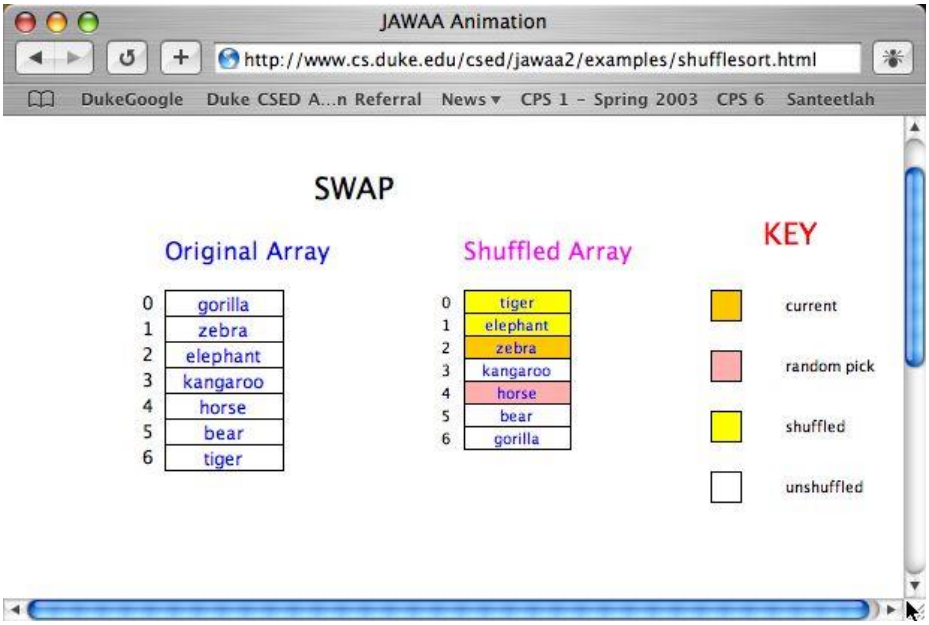



**Σχήμα 3.10. Η ταξινόμηση της φυσαλίδας στο Animal**

Επειδή οι οπτικοποιήσεις που παράγονται είναι γραμμένες σε Java μπορούν να εκτελεστούν σε οποιοδήποτε υπολογιστή και λειτουργικό σύστημα, δηλαδή είναι ανεξάρτητες πλατφόρμας. Το σημαντικό όμως μειονέκτημα είναι ότι δεν απευθύνεται τόσο σε μαθητές/φοιτητές όσο σε διδάσκοντες, αφού για να μπορεί να τη χρησιμοποιήσει κάποιος θα πρέπει ήδη να γνωρίζει προγραμματισμό σε Java, μαζί με τη γλώσσα AnimalScript. Υπάρχει φυσικά ο γραφικός συντάκτης, ο οποίος όμως δεν έχει την πληρότητα και τη δύναμη έκφρασης που έχει η γλώσσα σεναρίων οπτικοποίησης AnimalScript.

### *3.3.5 JAWAA*

Το JAWAA (Java and Web-based Algorithm Animation) είναι ένα σύστημα οπτικοποίησης αλγορίθμων και δομών δεδομένων που αποτελείται από τρία τμήματα (Pierson & Rodger, 1998): α) τη γλώσσα περιγραφής οπτικοποιήσεων, β) το γραφικό συντάκτη και γ) ένα applet, το οποίο εκτελεί τις οπτικοποιήσεις (Σχήμα 3.11). Μοιάζει σε αρκετά σημεία με το JSamba και το Animal αφού ο χρήστης μπορεί να κατασκευάσει μια οπτικοποίηση είτε με τη βοήθεια της γραφικής διεπαφής ή με την ειδική γλώσσα σεναρίων του συστήματος. Το μεγάλο πλεονέκτημα είναι η ανεξαρτησία πλατφόρμας, αφού το σύστημα είναι γραμμένο σε Java και η πρόσβαση σε αυτό μέσω του Παγκόσμιου Ιστού γίνεται εύκολα και γρήγορα, αφού αρκεί να αναρτήσουμε το applet στην ιστοσελίδα που θέλουμε.




**Σχήμα 3.11. Οπτικοποίηση του αλγόριθμου ταξινόμησης στο Jawaa**

Το σύστημα υποστηρίζει την οπτικοποίηση δομών δεδομένων, όπως είναι η στοίβα, η ουρά, το δέντρο, ο γράφος, και η λίστα, μαζί με τις αντίστοιχες λειτουργίες τους. Ωστόσο, αν θέλουμε να κατασκευάσουμε την οπτικοποίηση του αλγορίθμου από τον κώδικά του, θα πρέπει να χρησιμοποιήσουμε τη γλώσσα συγγραφής σεναρίων της εφαρμογής, κάτι το οποίο θα ήταν αρκετά δύσκολο για αρχάριους προγραμματιστές.

### *3.3.6 MatrixPro και Trakla 2*

Το MatrixPro είναι ένα σύστημα οπτικοποίησης αλγορίθμων και δομών δεδομένων (Karavirta et al., 2004). Πρόκειται για μετεξέλιξη του συστήματος οπτικοποίησης Matrix (Korhonen & Malmi, 2002). Οι μαθητές μπορούν να πειραματιστούν με τις οπτικοποιήσεις διάφορων αλγορίθμων πάνω σε έτοιμες δομές δεδομένων που παρέχει το λογισμικό. Εκτός από τις βασικές δομές δεδομένων, υπάρχουν και υλοποιήσεις πιο πολύπλοκων δομών, όπως αυτή του ισοζυγισμένου δέντρου (AVL). Ο χρήστης μπορεί να ορίσει ένα αντικείμενο αυτού του τύπου και με διαδοχικές εισαγωγές και διαγραφές κόμβων να έχει μια άμεση οπτικοποίηση όχι μόνο της δομής του αλλά και των αλγορίθμων εισαγωγής και διαγραφής δεδομένων.

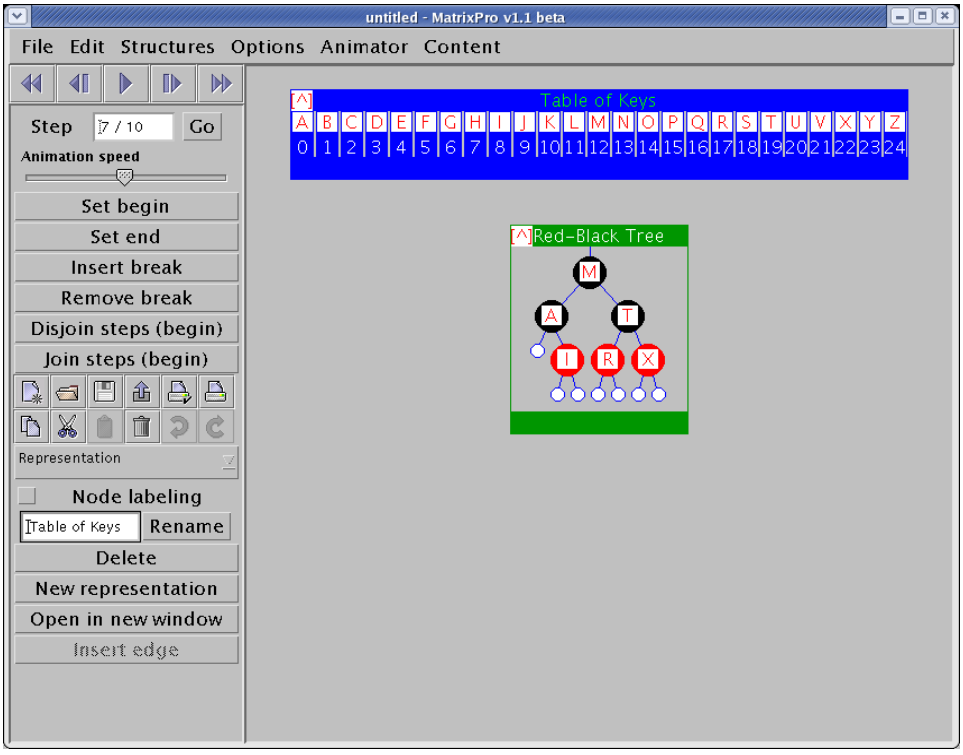
Οι σχεδιαστές του συστήματος έχουν αναπτύξει μια βάση ασκήσεων για τους μαθητές πάνω στο MatrixPro με στόχο την κατανόηση των βασικών λειτουργιών των δομών δεδομένων. Οι ασκήσεις αυτές αξιολογούνται αυτόματα από το σύστημα αξιολόγησης Trakla2 (Korhonen, Malmi & Silvasti, 2003). Το σύστημα αυτό είναι σχεδιασμένο πάνω από το MatrixPro και παρέχει υποστηρικτικό υλικό, όπως

σημειώσεις, διαλέξεις, επεξηγήσεις που εμπλουτίζονται συνεχώς. Έχει χρησιμοποιηθεί για την υποστήριξη ενός συνεργατικού διδακτικού μοντέλου με ασκήσεις οπτικοποίησης αλγορίθμων (Laakso, Myller & Korhonen, 2009). Τα MatrixPro και Trakla2 υποστηρίζουν μεγάλη ποικιλία δομών δεδομένων. Στο Σχήμα 3.12 δίνεται ένα παράδειγμα οπτικοποίησης μελανέρυθρων (red-black) δέντρων (Guibas & Sedgewick, 1978). Πρόκειται για μια ειδική περίπτωση ισοζυγισμένου δυαδικού δέντρου αναζήτησης που παρουσιάζει πολύ καλή απόδοση στις βασικές λειτουργίες του.

Θα πρέπει να αναφερθεί και το εκπαιδευτικό περιβάλλον Jype (Helminen & Malmi, 2010), το οποίο επιτρέπει την οπτικοποίηση προγραμμάτων γραμμένων στη γλώσσα Python, είναι βασισμένο στο MatrixPro.

### *3.3.7 Ville*

Μια άλλη ερευνητική ομάδα από το Πανεπιστήμιο Turku της Φινλανδίας έχει αναπτύξει το σύστημα οπτικοποίησης Ville (Rajala et al., 2007). Το εργαλείο αυτό υποστηρίζει την οπτικοποίηση προγραμμάτων (και όχι αλγορίθμων) σε διάφορες γλώσσες προγραμματισμού όπως C, C#, Python. Ο μαθητής επιλέγει τη γλώσσα προγραμματισμού, στην οποία θα κωδικοποιήσει έναν αλγόριθμο, και παρακολουθεί την οπτικοποίηση των προγραμματιστικών δομών κατά την εκτέλεση.




**Σχήμα 3.12. Η εισαγωγή σε μελανέρυθρα δέντρα στο MatrixPro**

Το εργαλείο υποστηρίζει τη δημιουργία και αυτοματοποιημένη αξιολόγηση των ασκήσεων των μαθητών συνεργαζόμενο με το Trakla2. Για το λόγο αυτό, αναφέρεται σε αυτή την κατηγορία, παρόλο που δεν θεωρείται σύστημα 'καθαρής' οπτικοποίησης αλγορίθμων.

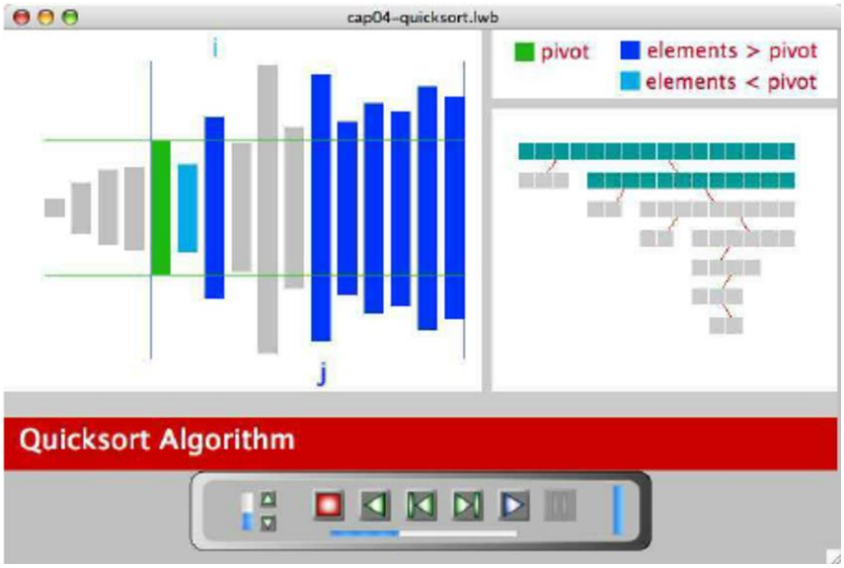
Η μελέτη του ίδιου παραδείγματος προγραμματισμού σε διάφορες γλώσσες, έχει το πλεονέκτημα ότι ο μαθητής αναγνωρίζει τις κοινές προγραμματιστικές δομές μεταξύ των γλωσσών, εντοπίζοντας τις ομοιότητες. Με τη μέθοδο της αφαίρεσης, ο μαθητής επικεντρώνεται στη μελέτη των βασικών χαρακτηριστικών του αλγορίθμου χωρίς να χάνει χρόνο με τις συντακτικές και τεχνικές λεπτομέρειες της δομής κάθε γλώσσας.

### *3.3.8 Alvie*

Το Alvie είναι ένα σύστημα οπτικοποίησης αλγορίθμων που περιέχει πολλές έτοιμες οπτικοποιήσεις, από απλούς αλγορίθμους ταξινόμησης μέχρι και αλγορίθμους γράφων (Crescenzi & Nocentini, 2007). Πρόσφατα οι δημιουργοί του, σχεδίασαν οπτικοποιήσεις των αποδείξεων για την NP-πληρότητα γνωστών προβλημάτων, όπως αυτό της κάλυψης κορυφών και των 3 χρωμάτων. Το σύστημα έχει μια πολύ απλή διεπαφή και αποτελεί μέρος ενός ηλεκτρονικού βιβλίου το οποίο περιέχει τις περιγραφές όλων των αλγορίθμων μαζί με τις οπτικοποιήσεις τους. Για τη δημιουργία νέων οπτικοποιήσεων χρησιμοποιείται μία XML γλώσσα σεναρίων.

### *3.3.9 Leonardo Web*

Το σύστημα Leonardo Web (Bonifaci et al., 2006) είναι γραμμένο σε Java και διατίθεται με τη μορφή applet. Έτσι μπορεί κανείς να το κατεβάσει και να το εκτελέσει στον υπολογιστή του. Η οπτικοποίηση του αλγορίθμου δεν γίνεται αυτόματα. Ο χρήστης είναι υποχρεωμένος να χρησιμοποιήσει τη γλώσσα συγγραφής σεναρίων Alpha, να ορίσει γραφικά αντικείμενα και να τα αντιστοιχίσει σε μεταβλητές. Οι αλλαγές των μεταβλητών οπτικοποιούνται μέσω των γραφικών αντικειμένων. Η αντιστοίχιση αυτή ολοκληρώνεται με την προσθήκη κατάλληλων εντολών της γλώσσας Alpha στις εντολές του κώδικα του προγράμματος. Ουσιαστικά η γλώσσα Alpha προσθέτει στον κώδικα τις εντολές οπτικοποίησης. Στη συνέχεια γίνεται η μείξη των εντολών οπτικοποίησης με τον κώδικα του προγράμματος, ώστε να προκύψει ένα είδος ενδιάμεσου-αντικείμενου κώδικα.



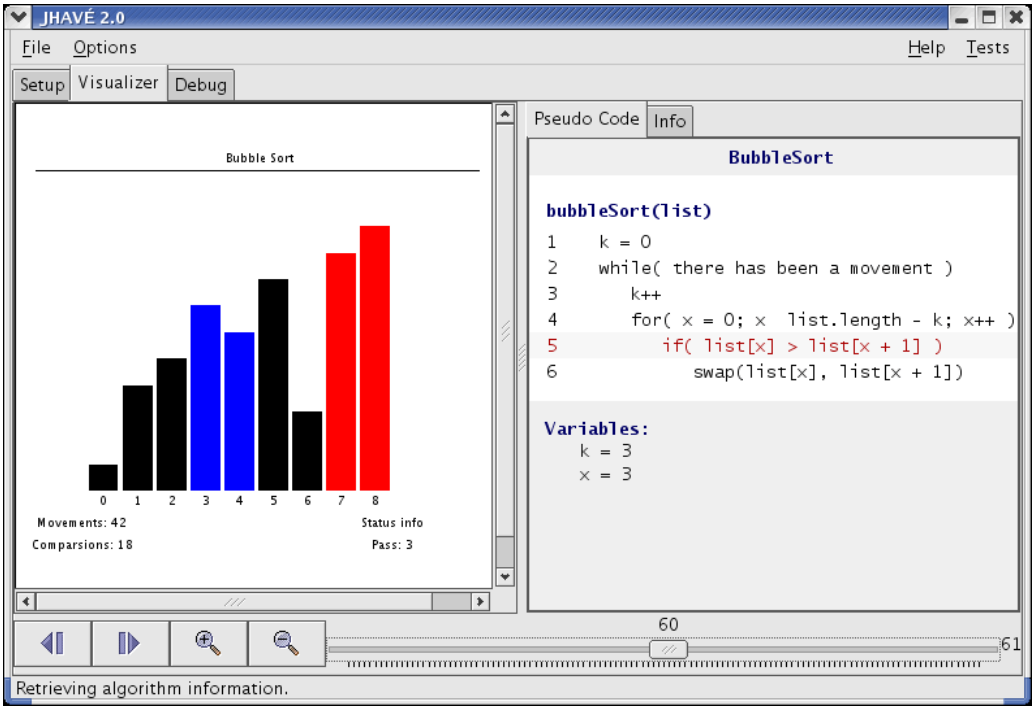

**Σχήμα 3.13. Η οπτικοποίηση της γρήγορης ταξινόμησης στο Leonardo Web**

Ο κώδικας αυτός εκτελείται από μια ιδεατή μηχανή υπολογισμού (virtual CPU). Η ιδεατή μηχανή προσφέρει τη δυνατότητα αντίστροφης εκτέλεσης του αλγορίθμου και της οπτικοποίησης του. Η δυνατότητα αυτή υλοποιείται σε πολύ λίγα περιβάλλοντα οπτικοποίησης προγραμμάτων.

### *3.3.10 JHave*

Το JHave (Java Hosted Algorithm Visualization Environment) (Naps, Eagan & Norton, 2003) δεν θεωρείται ένα σύστημα οπτικοποίησης αλγορίθμων από μόνο του, αλλά ένα διαδραστικό περιβάλλον παρουσίασης οπτικοποιήσεων αλγορίθμων που έχουν δημιουργηθεί από άλλα συστήματα. Το JHave βασίζεται στην αρχιτεκτονική πελάτη-διακομιστή (client-server) και έχει υλοποιηθεί σε Java.

Ο διακομιστής διαχειρίζεται τους διαθέσιμους αλγορίθμους οπτικοποίησης και δημιουργεί σενάρια που περιγράφουν την οπτικοποίηση του αλγορίθμου που ζητήθηκε από τον πελάτη. Η εφαρμογή-πελάτης εκτελείται στον υπολογιστή του χρήστη, ο οποίος επιλέγει έναν από τους διαθέσιμους αλγορίθμους. Στη συνέχεια ο πελάτης αποστέλλει μια αίτηση στον διακομιστή και τον ενημερώνει για τον επιλεγμένο αλγόριθμο.




**Σχήμα 3.14. Η οπτικοποίηση της ταξινόμησης φυσαλίδας στο JHave**

Η εφαρμογή-διακομιστής παράγει την οπτικοποίηση του αλγορίθμου και την αποθηκεύει σε ένα αρχείο με τη μορφή μιας γλώσσα σεναρίων (script language). Στη συνέχεια το αρχείο αυτό αποστέλλεται στο java applet του πελάτη, το οποίο είναι έτσι σχεδιασμένο ώστε να οπτικοποιεί αρχεία γραμμένα στη γλώσσα αυτή. Με την αρχιτεκτονική αυτή το JHave μπορεί να υποστηρίζει πολλές γλώσσες σεναρίων, αφού η μηχανή παρουσίασης της οπτικοποίησης είναι ανεξάρτητη της εφαρμογής που κατασκευάζει το αρχείο περιγραφής της οπτικοποίησης του αλγορίθμου. Στην παρούσα μορφή του, το JHave υποστηρίζει τη γλώσσα GAIGS (Generalized Algorithm Illustration through Graphical Software), και τις γλώσσες Animal και AnimalScript (Rössling & Freisleben, 2002). Ένα σημαντικό πλεονέκτημα του συστήματος είναι η δυνατότητα δημιουργίας ερωτήσεων κλειστού τύπου κατά την εκτέλεση της οπτικοποίησης.

### *3.3.11 HalVis*

Το σύστημα οπτικοποίησης αλγορίθμων HalVis (Hypermedia Algorithm Visualization), αναπτύχθηκε στο πανεπιστήμιο Auburn των Ηνωμένων Πολιτειών (Hansen, Narayanant & Hegarty, 2002). Πρόκειται για μια καθαρά υπερμεσική εφαρμογή, που υλοποιήθηκε με το λογισμικό συγγραφής εφαρμογών πολυμέσων Toolbook.


**Σχήμα 3.15. Εισαγωγή στην ταξινόμηση με επιλογή στο HALVIS**

Το σύστημα παρέχει διαφορετικές όψεις της εκτέλεσης του ίδιου αλγορίθμου. Ο χρήστης μπορεί να έχει ανοικτά τέσσερα παράθυρα: α) το παράθυρο ψευδοκώδικα, στο οποίο επισημαίνεται η εντολή που εκτελείται, β) το παράθυρο με τη γραφική αναπαράσταση των δομών δεδομένων, γ) το παράθυρο στο οποίο φαίνονται οι τιμές των μεταβλητών, και δ) το παράθυρο με επεξηγήσεις για κάθε βήμα του αλγορίθμου. Επίσης υιοθετούνται κατάλληλες αναπαραστάσεις των αντικειμένων, ώστε να τονίζονται τα ιδιαίτερα χαρακτηριστικά κάθε αλγορίθμου. Έτσι, κάθε αλγόριθμος ταξινόμησης μπορεί να έχει διαφορετική οπτικοποίηση.

Ένα άλλο σημαντικό πλεονέκτημα είναι η δημιουργία ερωτήσεων στους μαθητές κατά την εκτέλεση του αλγορίθμου. Το HalVis είναι ένα από τα πρώτα συστήματα που εισήγαγε αυτή την τεχνική των ερωταποκρίσεων κατά την εκτέλεση του αλγορίθμου.

Παρόλο που δεν προσφέρει την δυνατότητα στον μαθητή να υλοποιήσει τον δικό του αλγόριθμο, αποτελεί μια πολύ καλή επιλογή για μια πρώτη επαφή με τη λογική βασικών αλγορίθμων.

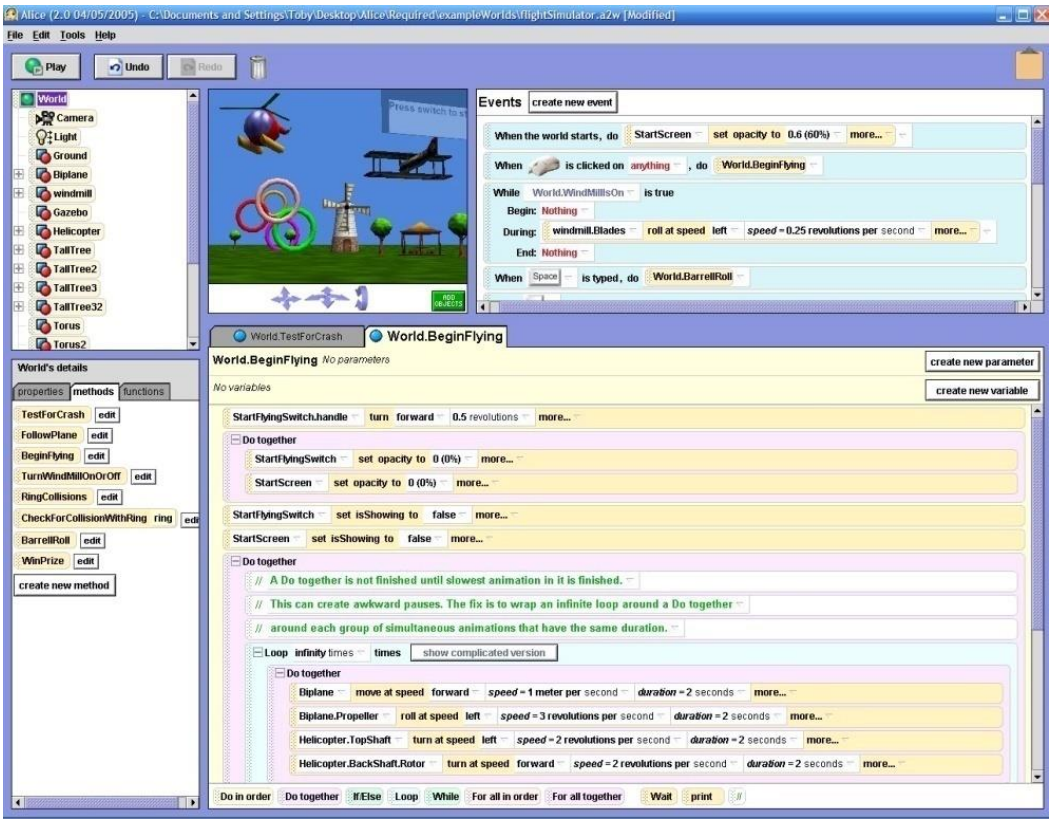
### *3.3.12 Alice*

Το Alice είναι ένα περιβάλλον προγραμματισμού που επιτρέπει τη χρήση τρισδιάστατων γραφικών και τη δημιουργία κινούμενης εικόνας με στόχο την αφήγηση μιας ιστορίας ή τη δημιουργία ενός διαδραστικού παιχνιδιού ή ενός βίντεο (Pausch

et.al., 1995). Επιτρέπει τη διαχείριση τρισδιάστατων αντικειμένων (π.χ. άνθρωποι, ζώα, οχήματα) σε έναν εικονικό κόσμο που δημιουργεί ο ίδιος ο μαθητής. Πρόκειται για ένα ελεύθερο διδακτικό εργαλείο, σχεδιασμένο για μια πρώτη επαφή με τον αντικειμενοστρεφή προγραμματισμό, μέσα από τη δημιουργία ταινιών κινουμένων σχεδίων, καθιστώντας τη διαδικασία της μάθησης πολύ ελκυστική για τους μαθητές.

Αν και δεν πρόκειται για ένα περιβάλλον που σχεδιάστηκε με αποκλειστικό στόχο την οπτικοποίηση αλγορίθμων, μπορεί να χρησιμοποιηθεί για το σκοπό αυτό ακόμη και από μικρούς μαθητές. Ο μαθητής καλείται να ορίσει τα αντικείμενα του ιδεατού κόσμου και τον τρόπο, με τον οποίο αυτά αντιδρούν στα διάφορα ερεθίσματα. Για την κωδικοποίηση του αλγορίθμου παρέχονται όλες οι γνωστές αλγοριθμικές δομές (ακολουθίας, επιλογής, επανάληψης) με την μορφή μπλοκ, όπως στο Scratch (2010).

Ο μαθητής επιλέγει αντικείμενα με το ποντίκι, οπότε αποφεύγει συντακτικά λάθη αξιοποιώντας το χρόνο του στην ανάπτυξη του αλγορίθμου και στην εμβάθυνση σε βασικές προγραμματιστικές δομές και έννοιες.



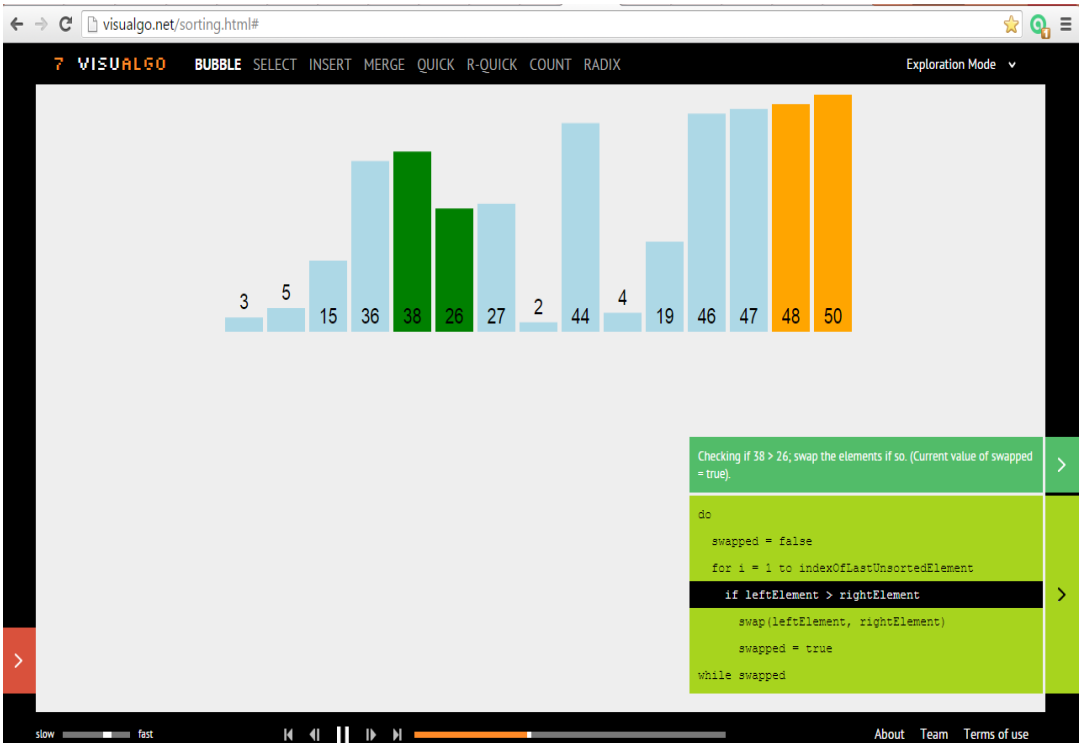

**Σχήμα 3.16. Το περιβάλλον προγραμματισμού Alice**

### *3.3.13 Σύγχρονα συστήματα οπτικοποίησης αλγορίθμων*

Τα τελευταία χρόνια με την διάδοση της τεχνολογίας HTML5 για την ανάπτυξη εφαρμογών που εκτελούνται απευθείας μέσα από ιστοσελίδες, έχει εμφανιστεί μια νέα γενιά περιβαλλόντων οπτικοποίησης. Τα περιβάλλοντα αυτά συνήθως δεν απαιτούν την εγκατάσταση πρόσθετων όπως απαιτούσαν τα java applets ή άλλες τεχνολογίες. Αρκεί κάποιος να έχει πρόσβαση σε κάποιο πρόγραμμα περιήγησης στον παγκόσμιο ιστό. Αυτό σημαίνει ότι τα περιβάλλοντα αυτά εκτελούνται όχι μόνο σε διαφορετικά λειτουργικά συστήματα (Windows, Linux, MacOS, Unix, Android, ChromeOS) αλλά και σε όσες υπολογιστικές πλατφόρμες (επιτραπέζιοι υπολογιστές, ταμπλέτες, νέας γενιάς κινητά) προσφέρουν πρόσβαση στον παγκόσμιο ιστό μέσα από κάποιο πρόγραμμα περιήγησης (browser).

Έτσι τα περιβάλλοντα αυτά είναι συνήθως δυναμικές ιστοσελίδες οι οποίες εκτελούν κάποιον κώδικα στον υπολογιστή του χρήστη (client-based) και είναι βασισμένα σε τεχνολογία HTML5/Javascript.

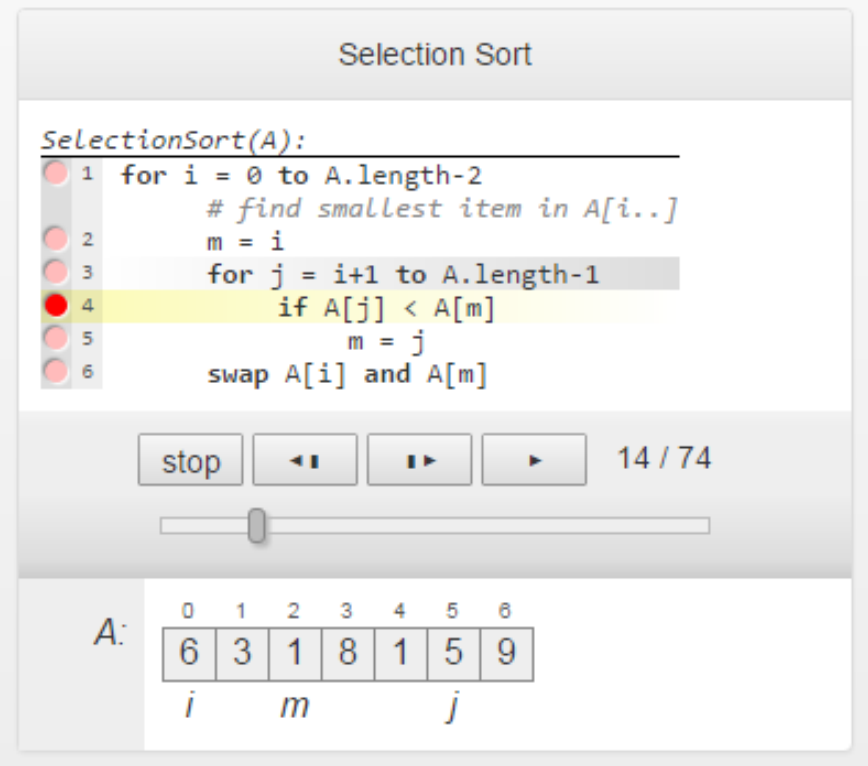
Ένα τέτοιο περιβάλλον είναι το VisuAlgo (http://visualgo.net) (Halim, 2015) το οποίο είναι ίσως το πληρέστερο σύστημα οπτικοποίησης αλγορίθμων αφού υποστηρίζει σχεδόν όλες τις κατηγορίες γνωστών αλγορίθμων όπως αλγόριθμοι ταξινόμησης και αναζήτησης σε πίνακες, αλγόριθμοι γράφων, αλγόριθμοι εισαγωγής/διαγραφής σε διάφορες δομές δεδομένων όπως στοίβα, ουρά, δέντρα, σωρός κ.α. Το σύστημα επιτρέπει στον χρήστη να πειραματιστεί με κάθε αλγόριθμο εισάγοντας τα δικά του δεδομένα. Το σύστημα ξεκίνησε ως μια προσπάθεια να κατασκευαστούν επεξηγηματικές οπτικοποιήσεις αλγορίθμων για μαθητές που συμμετέχουν σε διαγωνισμούς και ολυμπιάδες πληροφορικής. Το σύστημα δέχεται πάνω από 2000 επισκέψεις σε καθημερινή βάση. Διαθέτει δύο καταστάσεις λειτουργίας, εκμάθησης (tutorial mode) και εξερεύνησης (exploration mode). Στην πρώτη κατάσταση εμφανίζονται επεξηγηματικά αναδυόμενα παράθυρα στον χρήστη σε κάθε βήμα του αλγορίθμου. Εκτός από τις επεξηγήσεις το λογισμικό παρέχει και ερωτήσεις για κάθε αλγόριθμο με τη μορφή ενός κουίζ. Οι απαντήσεις του μαθητή αξιολογούνται άμεσα από το σύστημα, οπότε η διαδικασία αυτή μπορεί να χρησιμοποιηθεί ως μια δοκιμασία αυτοαξιολόγησης της γνώσης των μαθητών για συγκεκριμένες δομές δεδομένων και αλγορίθμους. Παράλληλα με την οπτικοποίηση φαίνεται και ο αλγόριθμος σε μορφή ψευδοκώδικα με επισήμανση της εντολής που εκτελείται σε κάθε βήμα. Ωστόσο ο χρήστης δεν μπορεί να τροποποιήσει τον κώδικα.




**Σχήμα 3.17. Η ταξινόμηση της φυσαλίδας στο περιβάλλον VisuAlgo**

Ένα άλλο δημοφιλές περιβάλλον οπτικοποίησης αλγορίθμων που ανήκει στην ίδια κατηγορία είναι το Vamonos (Carmer & Rosulek, 2015). Όπως και στο VisuAlgo έτσι και στο Vamonos οι οπτικοποιήσεις εκτελούνται στον υπολογιστή του χρήστη μέσα από μια ιστοσελίδα. Το περιβάλλον αυτό παρέχει τη δυνατότητα της ενσωμάτωσης των οπτικοποιήσεων σε άλλες ιστοσελίδες ή ψηφιακά βιβλία. Το σύστημα έχει και αυτό υψηλό βαθμό διαδραστικότητας όπως και το VisualAlgo αφού επιτρέπει εισαγωγή δεδομένων από τον χρήστη και τον πειραματισμό με την εκτέλεση των οπτικοποιήσεων. Το λογισμικό υποστηρίζει σε κάποιο βαθμό δυνατότητες εκσφαλμάτωσης όπως τον ορισμό σημείων διακοπής και την παρακολούθηση μεταβλητών, παράλληλα με την οπτικοποίηση του αλγορίθμου. Και αυτό το λογισμικό υποστηρίζει μια μεγάλη γκάμα αλγορίθμων όπως αλγόριθμους δυναμικού προγραμματισμού, ελάχιστων συνδετικών δέντρων, μέγιστης ροής κλπ.

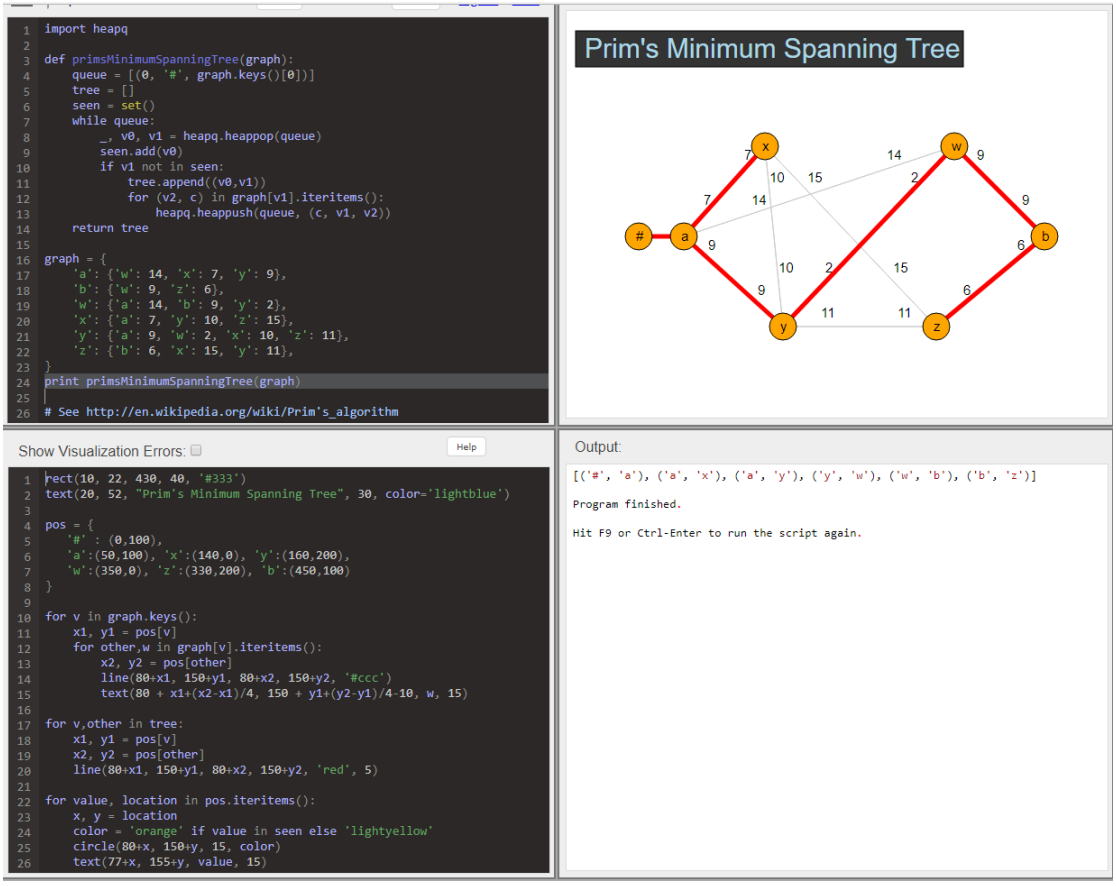
Επίσης και στα δύο παραπάνω συστήματα ο χρήστης μπορεί να επιλέξει παύση της οπτικοποίησης ή εκτέλεση βήμα-βήμα, ώστε να επικεντρωθεί στα κρίσιμα σημεία του αλγορίθμου. Και σε αυτό το περιβάλλον ο χρήστης δεν μπορεί να τροποποιήσει τον κώδικα του αλγορίθμου.




**Σχήμα 3.18. Η ταξινόμηση με επιλογή στο περιβάλλον οπτικοποίησης Vamonos**

Το τρίτο περιβάλλον οπτικοποίησης που ανήκει σε αυτή την κατηγορία είναι το **PyAlgoViz** (Laffra, 2014) το οποίο είναι επίσης γραμμένο σε HTML5 και εκτελείται μέσα από μια ιστοσελίδα (http://pyalgoviz.appspot.com/). Επιτρέπει την κωδικοποίηση ενός αλγορίθμου στην γλώσσα προγραμματισμού Python και την οπτικοποίησή του. Η διαφορά με τα προηγούμενα δύο περιβάλλοντα είναι ότι εδώ δεν γίνονται όλα στην πλευρά του χρήστη αλλά σημαντικό μέρος της δουλειάς λαμβάνει χώρα σε έναν απομακρυσμένο διακομιστή (server) της Google, το Google App Engine.

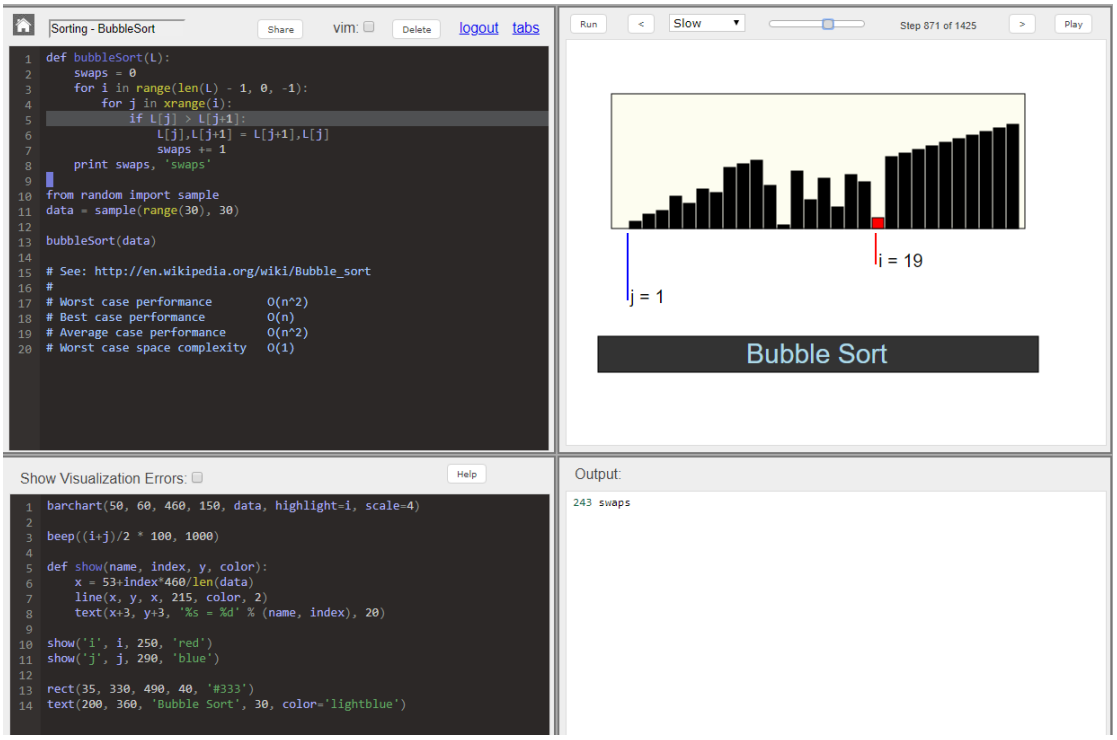
Το PyAlgoViz επιτρέπει την τροποποίηση του κώδικα του αλγορίθμου ο οποίος δεν δίνεται σε ψευδοκώδικα αλλά είναι γραμμένος στην γλώσσα προγραμματισμού Python. Ο κώδικας αυτός εκτελείται από τον διακομιστή ο οποίος αφού ελέγξει τον κώδικα, τον εκτελεί και παράγει την οπτικοποίησή του. Στη συνέχεια ο διακομιστής στέλνει την οπτικοποίηση στον υπολογιστή του χρήστη όπου τρέχει το αντίστοιχο πρόγραμμα-πελάτης το οποίο αναπαράγει την οπτικοποίηση. Η οπτικοποίηση είναι απλή και χαρακτηρίζεται από χαμηλής ποιότητας γραφικά. Η αναπαράσταση που έχει επιλεγεί δείχνει τα στοιχεία του πίνακα σαν μπάρες διαφορετικού ύψους ώστε να είναι ορατή η διαφορά κατά τη σύγκριση γειτονικών στοιχείων.




**Σχήμα 3.19. Ο αλγόριθμος του Prim για τον υπολογισμό του ελάχιστου συνδετικού δέντρου στο περιβάλλον PyAlgoViz**

Το λογισμικό προϋποθέτει ότι ο μαθητής/φοιτητής έχει ήδη καλή γνώση της γλώσσας προγραμματισμού Python. Διαθέτει μεγάλη ποικιλία οπτικοποιήσεων οι οποίες δεν αναφέρονται μόνο σε αλγορίθμους αλλά και σε μαθηματικές έννοιες, όπως π.χ. είναι οι πρώτοι αριθμοί.

Ένα σοβαρό μειονέκτημα του συστήματος είναι ότι οι γραφικές αναπαραστάσεις που έχουν επιλεγεί δεν είναι τόσο επεξηγηματικές όσο οι αναπαραστάσεις άλλων συστημάτων. Επίσης η επικοινωνία με τον διακομιστή σε κάποιες περιπτώσεις μπορεί να καθυστερήσει λίγο ανάλογα με τον φόρτο. Ωστόσο το γεγονός ότι πρόκειται για το μοναδικό σύστημα που επιτρέπει την τροποποίηση του κώδικα και την άμεση οπτικοποίηση του αλγορίθμου είναι ένα πολύ μεγάλο πλεονέκτημα.




**Σχήμα 3.20. Η ταξινόμηση ευθείας ανταλλαγής στο περιβάλλον PyAlgoViz**

### *3.3.14 Το Φωτόδεντρο*

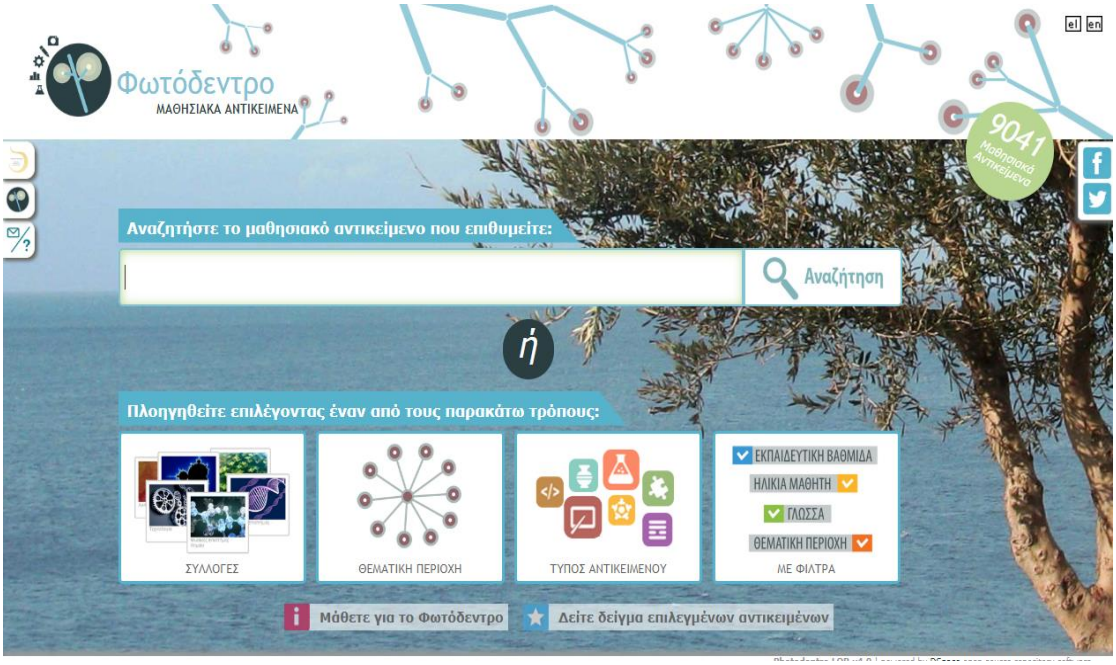
Τα τελευταία χρόνια παρατηρείται διεθνώς μεγάλο εκπαιδευτικό ενδιαφέρον για τους Ανοικτούς Εκπαιδευτικούς Πόρους (Open Educational Resources, OER), οι οποίοι είναι ελεύθερα διαθέσιμοι μέσω του Διαδικτύου για χρήση στους εκπαιδευόμενους (μαθητές, σπουδαστές) και στους διδάσκοντες. Τα αποθετήρια ανοικτών εκπαιδευτικών πόρων, όπως το MERLOT (www.merlot.org), το PheT (https://phet.colorado.edu) κ.α., συγκεντρώνουν και ενοποιούν σημασιολογικά έναν μεγάλο όγκο μαθησιακών αντικειμένων και ψηφιακού εκπαιδευτικού υλικού και παρέχουν λεπτομερείς περιγραφές μεταδεδομένων, ώστε να διευκολύνουν την αναζήτηση και την πρόσβαση σε αυτά από διδάσκοντες και μαθητές (Τζιμογιάννης, 2017).

Το Φωτόδεντρο (http://photodentro.edu.gr) είναι το ελληνικό αποθετήριο μαθησιακών αντικειμένων για την πρωτοβάθμια και δευτεροβάθμια εκπαίδευση, το οποίο αναπτύχθηκε στο πλαίσιο του έργου Ψηφιακό Σχολείο. Αποτελεί την κεντρική e-υπηρεσία του Υπουργείου Παιδείας για την ενοποιημένη αναζήτηση και διάθεση ψηφιακού εκπαιδευτικού περιεχομένου στα σχολεία (Megalou & Kaklamanis, 2014). Το Φωτόδεντρο προωθεί τη χρήση των ανοιχτών εκπαιδευτικών πόρων στα σχολεία, υλοποιώντας την εθνική στρατηγική για το ψηφιακό εκπαιδευτικό περιεχόμενο. Όλο

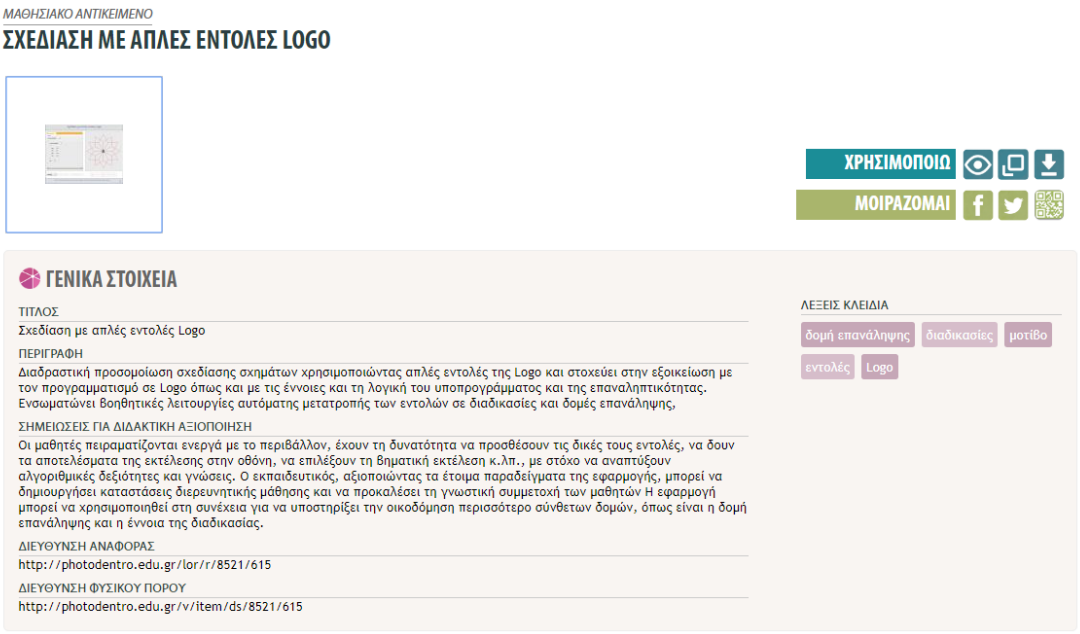
το υλικό που είναι διαθέσιμο διατίθεται ελεύθερα, με την άδεια Creative Commons CC BY-NC-SA ή με άλλη παρόμοια, πιο ανοιχτή άδεια χρήσης.

Σήμερα το αποθετήριο περιέχει 9041 μαθησιακά αντικείμενα για όλα τα γνωστικά αντικείμενα που διδάσκονται στην πρωτοβάθμια και δευτεροβάθμια εκπαίδευση (Σχήμα 3.21). Πολλά από τα μαθησιακά αντικείμενα είναι ενσωματωμένα στα εμπλουτισμένα ηλεκτρονικά σχολικά βιβλία. Ο χρήστης μπορεί να αναζητήσει το μαθησιακό αντικείμενο που θέλει με βάση διάφορα κριτήρια, όπως θέμα, βαθμίδα εκπαίδευσης και άλλα.

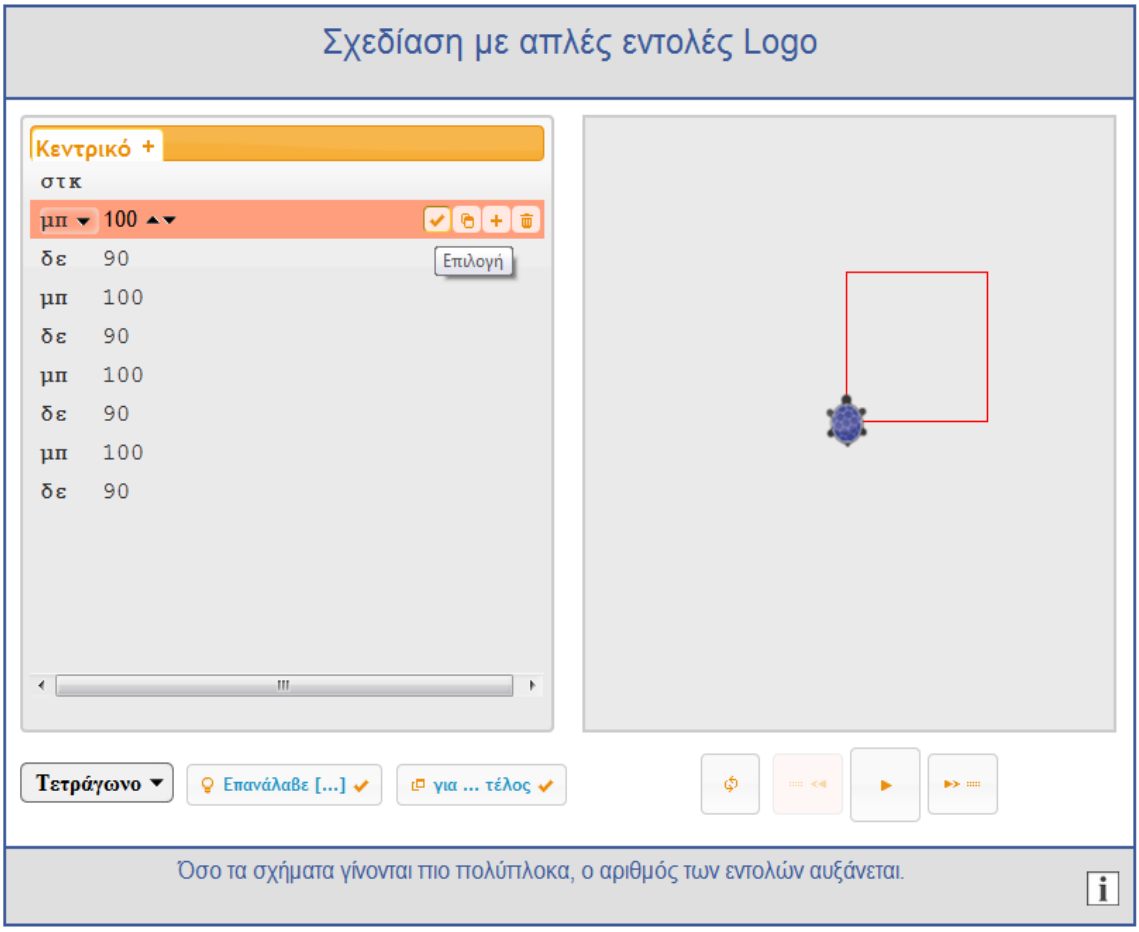
Το Φωτόδεντρο περιλαμβάνει επίσης μαθησιακά αντικείμενα Πληροφορικής και προγραμματισμού υπολογιστών (Jimoyiannis et al., 2013). Στο Σχήμα 3.22 φαίνεται η αρχική οθόνη του μαθησιακού αντικειμένου με τίτλο "Σχεδίαση με απλές εντολές LOGO" με τις πληροφορίες μεταδεδομένων. Το Σχήμα 3.23 δείχνει την προσομοίωση της διαδικασίας σχεδίασης τεραγώνου στη LOGO που υποστηριζει το μαθησιακό αντικείμενο. Ο μαθητής μπορεί να πληκτρολογήσει ένα πρόγραμμα σε LOGO και να παρατηρήσει άμεσα το αποτέλεσμα της εκτέλεσής του. Μπορεί να κατεβάσει το μαθησιακό αντικείμενο στον υπολογιστή του ή να το ενσωματώσει σε μια δική τους ιστοσελίδα.




**Σχήμα 3.21. Η αρχική ιστοσελίδα του Φωτόδεντρου**




**Σχήμα 3.22. Περιγραφή μεταδεδομένων του μαθησιακού αντικειμένου «Σχεδίαση με απλές εντολές LOGO»**




**Σχήμα 3.23. Εκτέλεση μαθησιακού αντικειμένου: Προσομοίωση της διαδικασίας σχεδίασης τετραγώνου στη LOGO**

### *3.3.15 Η Συνδυασμένη ταξινομία Price / Naps*

Τα συστήματα οπτικοποίησης προγραμμάτων αναπαριστούν γραφικά βασικές προγραμματιστικές δομές, όπως οι μεταβλητές, οι δείκτες και οι πίνακες που είναι χρήσιμες στους προγραμματιστές, ειδικά κατά την εκσφαλμάτωση (debugging) των προγραμμάτων. Μέσω της οπτικοποίησης γίνεται παρακολούθηση των τιμών των μεταβλητών σε κάθε βήμα του αλγορίθμου, με στόχο την ανίχνευση και τη διόρθωση των λαθών του προγράμματος.

Στη βιβλιογραφία αναφέρονται διάφορες ταξινομίες για την κατηγοριοποίηση λογισμικών οπτικοποίησης. Μια από τις πρώτες ήταν η ταξινομία του Myers (1990), στην οποία τα συστήματα κατηγοριοποιούνται με βάση δύο συνιστώσες: α) τον τύπο της οπτικοποίησης, δηλαδή τι οπτικοποιείται (κώδικας, δεδομένα, αλγόριθμος) και β) τη μορφή εμφάνισης, δηλαδή αν πρόκειται για στατική ή δυναμική οπτικοποίηση (με κίνηση).

Οι Stasko & Patterson (1992) πρότειναν τέσσερα κριτήρια ταξινόμησης: την όψη/πτυχή (aspect), το επίπεδο αφαίρεσης (abstraction), την οπτικοποίηση (animation) και το επίπεδο αυτοματισμού (automation).

Σε μια άλλη δημοφιλή ταξινομία οι Roman & Cox (1993) χρησιμοποίησαν πέντε κριτήρια κατηγοριοποίησης: την εμβέλεια(scope), το επίπεδο αφαίρεσης (abstraction), τη μέθοδο ανάπτυξης της οπτικοποίησης (specification method), και την τεχνική παρουσίασης (technique) που αναφέρεται στο σχεδιασμό της διεπαφής και στην αναπαράσταση της πληροφορίας.

Η ταξινομία του Price (Price, Baecker & Small, 1993) θεωρείται η πληρέστερη ταξινομία συστημάτων οπτικοποίησης και χρησιμοποιείται στην παρούσα συγκριτική μελέτη σε συνδυασμό με την ταξινομία του Naps (2003). Η ταξινομία του Price αναλύει τα συστήματα οπτικοποίησης αλγορίθμων-προγραμμάτων με βάση έξι κριτήρια-δείκτες: εμβέλεια (scope), περιεχόμενο (content), μορφή (form), μέθοδος (method), αποτελεσματικότητα (effectiveness), αλληλεπίδραση (interaction).

**Εμβέλεια (Scope)**: Η εμβέλεια περιγράφει το εύρος των προγραμμάτων ή αλγορίθμων που το σύστημα μπορεί να οπτικοποιήσει. Αν δηλαδή έχει σχεδιαστεί για την οπτικοποίηση συγκεκριμένων προγραμμάτων/αλγορίθμων ή μπορεί να οπτικοποιήσει οποιοδήποτε αλγόριθμο, ο οποίος ανήκει σε μία κλάση αλγορίθμων. Όσο ευρύτερη είναι αυτή η κλάση τόσο μεγαλύτερη θεωρείται η εμβέλεια του συστήματος.

Το σύστημα που ξεχωρίζει σε αυτή την κατηγορία είναι το jGrasp, κυρίως, επειδή επιτρέπει επίσης την οπτικοποίηση των νημάτων όσων προγραμμάτων χρησιμοποιούν παράλληλο προγραμματισμό. Επίσης, το MatrixPro περιέχει πολλές κατηγορίες οπτικοποίησης δομών δεδομένων, όπως πίνακες, λίστες, δέντρα, γραφήματα κλπ.

Η πλατφόρμα στην οποία εκτελείται κάθε σύστημα είναι ένας ακόμα δείκτης της εμβέλειας. Τα περισσότερα συστήματα είναι γραμμένα στη γλώσσα προγραμματισμού Java και έτσι μπορούν να εκτελεστούν σε όλα τα λειτουργικά συστήματα αλλά και μέσω του Παγκόσμιου Ιστού με τη μορφή applets (Animal, Jawaa).

**Περιεχόμενο (Content)**: Το περιεχόμενο αναφέρεται στις πτυχές του προγράμματος ή του αλγορίθμου που θέλουμε να οπτικοποιήσουμε. Τέτοιες πτυχές είναι ο κώδικας του προγράμματος, η αναπαράσταση των δεδομένων και γενικότερα οι αναπαραστάσεις που αναδεικνύουν τα βασικά χαρακτηριστικά της λειτουργίας του αλγορίθμου. Τα συστήματα χωρίζονται ουσιαστικά σε δύο κατηγορίες: α) συστήματα που οπτικοποιούν τις προγραμματιστικές (μεταβλητή, δείκτης, υποπρόγραμμα) ή τις αντικειμενοστρεφείς δομές (κλάση, αντικείμενο, πολυμορφισμός) και β) συστήματα που οπτικοποιούν τις αλγοριθμικές δομές, δηλαδή επιλέγουν κατάλληλες αναπαραστάσεις ώστε να αναδεικνύεται η λογική που κρύβεται πίσω από κάθε αλγόριθμο.

**Μορφή (Form)**: Το κριτήριο αυτό αφορά στα χαρακτηριστικά του αποτελέσματος της οπτικοποίησης, όπως το μέσο, τα γραφικά, οι αναπαραστάσεις και η κίνηση των αντικειμένων. Ένα πολύ σημαντικό χαρακτηριστικό είναι η εμφάνιση του κώδικα, ο οποίος εκτελείται συνήθως σε ένα παράθυρο με παράλληλη επισήμανση της τρέχουσα εντολής και του παραγόμενου γραφικού αποτελέσματος καθώς εκτελείται.

**Μέθοδος (Method)**: Το κριτήριο αυτό εξετάζει τον τρόπο με τον οποίο παράγεται η οπτικοποίηση. Σε κάποια συστήματα η παραγωγή της οπτικοποίησης γίνεται με αυτοματοποιημένο τρόπο από τον κώδικα του αλγορίθμου του χρήστη (Jeliot, jGrasp, Dave). Στα περισσότερα όμως, ο μαθητής πρέπει να κατασκευάσει ο ίδιος την οπτικοποίηση σε κάποια ειδική γλώσσα (Animal, Jawaa, JSamba). Αρκετά συστήματα παρέχουν τις δικές τους έτοιμες οπτικοποιήσεις προκαθορισμένων αλγορίθμων (MatrixPro, HalVis) με τις οποίες μπορεί να πειραματιστεί ο χρήστης. Το μοναδικό σύστημα οπτικοποίησης αλγορίθμων που προσφέρει τη δυνατότητα τροποποίησης του κώδικα και άμεσης οπτικοποίησής του είναι το PyAlgoViz, ωστόσο οι οπτικοποίησεις είναι χαμηλής ποιότητας και από πλευράς γραφικών αλλά και όσον αφορά την

πληροφορία που χρειάζεται ώστε να είναι επεξηγηματικές για έναν αρχάριο προγραμματιστή.

**Αποτελεσματικότητα (Effectiveness)**: Το κριτήριο αυτό εξετάζει πόσο αποτελεσματική είναι η οπτικοποίηση του αλγορίθμου τόσο σε τεχνολογικό όσο και σε εκπαιδευτικό ή μαθησιακό επίπεδο. Οι έρευνες που έχουν γίνει για αυτόν τον σκοπό είναι λίγες και δεν καλύπτουν όλα τα συστήματα, οπότε δεν μπορούν να βγουν ασφαλή συμπεράσματα. Για αυτό αποτελεί ένα ενδιαφέρον θέμα για μελλοντική έρευνα.

**Αλληλεπίδραση (Interaction)**: Το κριτήριο αυτό αξιολογεί το βαθμό εμπλοκής και αλληλεπίδρασης του εκπαιδευόμενου με το λογισμικό, σε παιδαγωγικό και γνωστικό επίπεδο. Κάποια λογισμικά θέτουν ερωτήσεις στο χρήστη, κατά την εκτέλεση της οπτικοποίησης, επιτρέπουν την παύση, την σταδιακή εκτέλεση βήμα-βήμα ή την εκτέλεση του αλγορίθμου προς τα πίσω, τον έλεγχο της ταχύτητας εκτέλεσης της προσομοίωσης, ακόμη και την κατασκευή της οπτικοποίησης του αλγορίθμου από τον ίδιο το χρήστη.

Στον Πίνακα 3.1 παρουσιάζονται συγκριτικά τα συστήματα οπτικοποίησης που μελετήθηκαν και τα βασικά τους χαρακτηριστικά, σύμφωνα με τους άξονες της ταξινομίας του Price, οι οποίοι αναλύονται στη συνέχεια.

Το κριτήριο της αλληλεπίδρασης αποτελεί τη σημαντικότερη παράμετρο για την παιδαγωγική αποτελεσματικότητα ενός συστήματος οπτικοποίησης, όπως δείχνουν πολλές έρευνες (Hundhausen, Douglas & Stasko, 2002· Naps et al., 2003· Urquiza–Fuentes & Velazquez–Iturbide, 2009· Velazquez-Iturbide, Hernan-Losada & Paredes-Velasco, 2017). Για την ανάλυση του βαθμού αλληλεπίδρασης χρησιμοποιείται η ταξινομία του Naps (2003), η οποία διακρίνει έξι επίπεδα αλληλεπίδρασης του λογισμικού με τον χρήστη. Η κεντρική ιδέα της ταξινομίας είναι ότι το επίπεδο αλληλεπίδρασης που υποστηρίζεται από το λογισμικό, έχει άμεση σχέση με τα εκπαιδευτικά οφέλη που αποκομίζει ο μαθητής από την ενασχόλησή του με αυτό.

Η κλίμακα ταξινομίας, όσον αφορά τα επίπεδα αλληλεπίδρασης των συστημάτων οπτικοποίησης, που θα χρησιμοποιήσουμε αναλύεται στη συνέχεια:

1. *Παρακολούθηση της οπτικοποίησης* (Viewing): Ο μαθητής απλά παρακολουθεί την οπτικοποίηση χωρίς να παρεμβαίνει. Όλα τα συστήματα οπτικοποίησης που εξετάσαμε υποστηρίζουν αυτή τη δυνατότητα.
2. *Ελεγχόμενη παρακολούθηση της οπτικοποίησης* (Controlled viewing): Ο μαθητής μπορεί να ελέγχει την ταχύτητα εμφάνισης της οπτικοποίησης στην

οθόνη ή να σταματάει εντελώς την εκτέλεση. Κάποια λογισμικά (Leonardo, jGrasp) παρέχουν ακόμη και τη δυνατότητα εκτέλεσης του αλγορίθμου με την ανάστροφη φορά (προς τα πίσω). Το επίπεδο της ελεγχόμενης παρακολούθησης αρχικά δεν υπήρχε στην ταξινομία του Naps αλλά προστέθηκε σε μια επέκτασή της που προτάθηκε πρόσφατα (Myller et al., 2009).

3. *Απόκριση σε ερωτήσεις* (Responding): Ο μαθητής απαντάει σε ερωτήσεις κατά την εκτέλεση του αλγορίθμου οι οποίες στις περισσότερες περιπτώσεις αξιολογούνται άμεσα (Trakla2, Ville). Έτσι ο μαθητής έχει τη δυνατότητα να διερευνήσει, να διορθώσει τα λάθη του και να προχωρήσει γνωστικά.
4. *Τροποποίηση* (Changing): Ο μαθητής μπορεί να αλλάξει τα δεδομένα εισόδου του αλγορίθμου ή να τροποποιήσει κάποια χαρακτηριστικά της οπτικοποίησης, όπως η ταχύτητα εκτέλεσης, η γραφική αναπαράσταση των αντικειμένων κ.λπ.
5. *Δημιουργία νέας οπτικοποίησης* (Constructing): Αφορά στις δυνατότητες του εκπαιδευόμενου να δημιουργήσει την δική του οπτικοποίηση. Υπάρχουν λογισμικά (π.χ. jGrasp, Jeliot, DAVE), όπου η οπτικοποίηση παράγεται αυτόματα από τον κώδικα του αλγορίθμου που συντάσσει ο ίδιος ο μαθητής. Στα περισσότερα συστήματα που μελετήθηκαν ο μαθητής είναι υποχρεωμένος να περιγράψει την οπτικοποίηση στην ειδική γλώσσα του συστήματος, η οποία είναι, συνήθως, μία γλώσσα σεναρίων (script language).

Από την ανάλυση που προηγήθηκε είναι προφανές ότι όσο υψηλότερο είναι το επίπεδο στην κλίμακα της ταξινομίας τόσο μεγαλύτερα εκπαιδευτικά οφέλη αναμένονται για το μαθητή. Τα τεχνολογικά και παιδαγωγικά χαρακτηριστικά, καθώς και τα αναπαραστατικά εργαλεία ενός συστήματος προσομοίωσης αλγορίθμων καθορίζουν τους τρόπους εκπαιδευτικής αξιοποίησης και χρήσης του λογισμικού από εκπαιδευτικούς και μαθητές.

**Πίνακας 3.1. Ταξινομία συστημάτων οπτικοποίησης κατά Price**

| Σύστημα | Εμβέλεια | Πλατφόρμα | Περιεχόμενο | Μορφή | Μέθοδος |
|---|---|---|---|---|---|
| Jeliot | Γλώσσα Java | Java | Αντικειμενοστρε-φείς δομές | Δομές Java, κώδικας | Αυτόματα |
| Leonardo | γλώσσα Alpha | Java applet | Προγραμματιστικές δομές | Δομές Δεδομένων | Αυτόματα |
| PlanAni | Pascal, C, κ.α. | Tcl/Tk | Ρόλοι μεταβλητών | Πολλαπλές όψεις | από τον χρήστη |
| WinHIPE | Συναρτησιακή γλώσσα Hope | Ανεξάρτητο πλατφόρμας | Συναρτησιακά χαρακτηριστικά | Αναδρομικές κλήσεις | Αυτόματα |
| JSamba | Ψευδογλώσσα | Java applet | Αλγοριθμικά χαρακτηριστικά | Αλγόριθμος, κώδικας | από τον χρήστη |
| Alvis Live | Cb/SALSA | .NET | Μεταβλητές, πίνακες | | Αυτόματα |
| Animal | Script γλώσσα | Java applet | Αλγοριθμικά χαρακτηριστικά | | από τον χρήστη |
| Jawaa | | | | | |
| MatrixPro | λίστες, δέντρα | Java | | Δομές Δεδομένων | Έτοιμες οπτικοποιήσεις |
| Trakla2 | Έτοιμες ασκήσεις | Java | Δομές Δεδομένων | Δομές Δεδομένων | Έτοιμες οπτικοποιήσεις |
| Ville | Java, C++, ψευδογλώσσα | Java | Προγραμματιστικές δομές | Εξήγηση για κάθε εντολή | Αυτόματα |
| JHave | Πλατφόρμα οπτικοποίησης | Java Web Start | Αλγοριθμικά χαρακτηριστικά | Αλγόριθμος | Έτοιμες οπτικοποιήσεις |
| Halvis | Μόνο αλγόριθμοι Ταξινόμησης | Toolbook | Αλγοριθμικά χαρακτηριστικά | Πολλαπλές αναπαραστάσεις | Έτοιμες οπτικοποιήσεις |
| Jive | Προγράμματα σε Java | Java | Αντικειμενοστρεφή χαρακτηριστικά | | Αυτόματα |
| jGrasp | | | | | |
| Alice | Ψευδογλώσσα | | Προγραμματιστικές δομές | Εικονικοί κόσμοι | |
| VisuAlgo | Ψευδογλώσσα | HTML5/ Javascript | Αλγοριθμικά Χαρακτηριστικά | Δομές Δεδομένων | Έτοιμες Οπτικοποιήσεις |
| Vamonos | | | | | |
| Thonny | | | | | Αυτόματα |
| PyAlgoViz | Python | | Προγραμματιστικές δομές | | |
| Python Tutor | | | | | |

**Πίνακας 3.2. Ταξινομία των συστημάτων οπτικοποίησης κατά Naps**

| Σύστημα | Παρακολούθηση | Ελεγχόμενη παρακολούθηση | Σύστημα Ερωτήσεων | Τροποποίηση Κώδικα | Δημιουργία οπτικοποίησης |
|---|---|---|---|---|---|
| Jeliot | ✓ | ✓ | - | ✓ | ✓ |
| Leonardo | ✓ | ✓ | | ✓ | - |
| PlanAni | ✓ | ✓ | | - | - |
| WinHIPE | ✓ | ✓ | | ✓ | ✓ |
| JSamba | ✓ | ✓ | - | ✓ | - |
| Alvis | ✓ | ✓ | | ✓ | ✓ |
| Animal | ✓ | ✓ | ✓ | ✓ | - |
| Jawaa | ✓ | ✓ | ✓ | ✓ | - |
| MatrixPro/ Trakla2 | ✓ | ✓ | ✓ | ✓ | |
| Ville | ✓ | ✓ | ✓ | ✓ | - |
| JHave | ✓ | ✓ | ✓ | - | |
| Halvis | ✓ | ✓ | ✓ | - | |
| Jive | ✓ | ✓ | ✓ | ✓ | ✓ |
| jGrasp | ✓ | ✓ | | ✓ | ✓ |
| Alice | ✓ | ✓ | | ✓ | ✓ |
| VisuAlgo | ✓ | ✓ | ✓ | | ✓ |
| Vamonos | ✓ | ✓ | | | ✓ |
| PyAlgoViz | ✓ | ✓ | | ✓ | ✓ |

✓ πλήρης λειτουργικότητα του συστήματος στο επίπεδο αυτό

- περιορισμένη λειτουργικότητα του συστήματος στο επίπεδο αυτό

Στον Πίνακα 3.2 παρουσιάζονται τα επίπεδα διαδραστικότητας των συστημάτων, σύμφωνα με την ταξινομία του Naps.

## *3.4 Συμπεράσματα*

Η δυσκολία παρουσίασης δυναμικών εννοιών όπως οι δομές δεδομένων και οι αλγόριθμοι επεξεργασίας τους, με συμβατικά μέσα, οδήγησε στην ανάπτυξη εναλλακτικών προσεγγίσεων και εκπαιδευτικών περιβαλλόντων οπτικοποίησης των εννοιών αυτών. Στο κεφάλαιο αυτό έγινε μια επισκόπηση και μια συγκριτική παρουσίαση των σημαντικότερων και πιο χαρακτηριστικών εκπαιδευτικών περιβαλλόντων οπτικοποίησης των αλγοριθμικών και προγραμματιστικών δομών. Χρησιμοποιήθηκε ο συνδυασμός των ταξινομιών Price και Naps και τα σχετικά κριτήρια περιγραφής για το βαθμό αλληλεπίδρασης που διαθέτει κάθε σύστημα. Έγινε μια ιστορική αναδρομή της εξέλιξης του επιστημονικού πεδίου με την παρουσίαση των

πρώτων συστημάτων οπτικοποίησης (Tango, Polka, Samba) και, στη συνέχεια, αναλύθηκαν τα χαρακτηριστικά σύγχρονων συστημάτων, όπως τα JSamba, Jawaa, Animal κ.α.).

Τα συστήματα αυτά μπορούν να βοηθήσουν τον εκπαιδευτικό να δημιουργήσει την οπτικοποίηση ενός αλγορίθμου, ώστε να υποστηρίξει τη διδασκαλία του στην τάξη. Το ζητούμενο όμως είναι, ακολουθώντας και τις διεθνείς τάσεις στο χώρο του εκπαιδευτικού λογισμικού, να διαμορφωθούν περιβάλλοντα τα οποία παρέχουν στο μαθητή δυνατότητες να πειραματιστεί με οπτικοποιήσεις αλγορίθμων, να ελέγξει τις αναπαραστάσεις του για αντικείμενα και δομές και, τέλος, να δημιουργήσει τις δικές του οπτικοποιήσεις. Ο σχεδιασμός των νεότερων συστημάτων οπτικοποίησης αλγορίθμων εστιάζει στις απαιτούμενες παιδαγωγικές προδιαγραφές και στο βαθμό αλληλεπίδρασης του συστήματος, ώστε να ενεργοποιούν και θα ενισχύουν το μαθητή. Στην κατεύθυνση αυτή οδήγησαν πολλές έρευνες σχετικά με την αποτελεσματικότητα των συστημάτων οπτικοποίησης (Hundhausen, Douglas & Stasko, 2002· Naps et al., 2003· Urquiza–Fuentes & Velazquez–Iturbide, 2009· Velazquez-Iturbide, Hernan-Losada & Paredes-Velasco, 2017) και έδειξαν ότι τα εκπαιδευτικά οφέλη για τους μαθητές καθορίζονται από τις χρησιμοποιούμενες αναπαραστάσεις και το βαθμό αλληλεπίδρασης που προσφέρει κάθε σύστημα.

Το ζήτημα των κατάλληλων γραφικών αναπαραστάσεων για τα υπολογιστικά αντικείμενα, ώστε να αναδεικνύονται για το μαθητή τα σημαντικά χαρακτηριστικά που διέπουν τη συμπεριφορά κάθε αλγορίθμου, αποτελεί την πρώτη παράμετρο μελέτης για το σχεδιασμό συστημάτων οπτικοποίησης. Το γενικό συμπέρασμα που προέκυψε από τη μελέτη των παραπάνω συστημάτων είναι ότι όσο περισσότερο αυτοματοποιημένη είναι η οπτικοποίηση ενός αλγορίθμου, τόσο λιγότερο προσαρμοσμένη είναι στα ιδιαίτερα χαρακτηριστικά του. Αντίθετα, τα συστήματα που προσφέρουν καλές αναπαραστάσεις έχουν χαμηλό βαθμό αυτοματοποίησης στην παραγωγή της οπτικοποίησης. Η δημιουργία μιας νέας οπτικοποίησης από τον ίδιο το χρήστη σε αυτά τα συστήματα, όταν αυτό επιτρέπεται, δεν είναι εύκολη υπόθεση για το μαθητή.

Το πρώτο επίπεδο αλληλεπίδρασης αναφέρεται στη δυνατότητα της ελεγχόμενης εκτέλεσης του αλγορίθμου με κάποιο γραφικό χειριστήριο, το οποίο επιτρέπει στο μαθητή να σταματά, να επιβραδύνει ή να επιταχύνει την εκτέλεση του αλγορίθμου, με βάση τις μαθησιακές ανάγκες του κάθε φορά. Στο δεύτερο επίπεδο αλληλεπίδρασης, το οποίο δεν υποστηρίζεται από όλα τα συστήματα, δίνεται η δυνατότητα στο μαθητή

να απαντήσει σε ερωτήσεις που γίνονται κατά την εκτέλεση του αλγορίθμου, όπως για παράδειγμα ‘τι θα συμβεί στο επόμενο βήμα;’ (HalVis, JHave). Κάποια συστήματα προσφέρουν επίσης άμεση διόρθωση των απαντήσεων των μαθητών πριν προχωρήσουν στο επόμενο βήμα (Trakla2, Ville). Το υψηλότερο επίπεδο αλληλεπίδρασης αφορά στις δυνατότητες που έχει ο μαθητής να δημιουργήσει την οπτικοποίηση του δικού του αλγορίθμου. Δηλαδή, να μπορεί να κωδικοποιήσει έναν αλγόριθμο σε κάποια γλώσσα προγραμματισμού, να δει άμεσα το αποτέλεσμα της οπτικοποίησής του και να αξιοποιήσει στη συνέχεια μέσα από κατάλληλα σενάρια (π.χ. ‘τι θα συμβεί αν αλλάξει κάποια υπολογιστή παράμετρος’). Υπάρχουν συστήματα που οπτικοποιούν αυτόματα τον αλγόριθμο του μαθητή, όπως το Jeliot, το Alvis ή το jGrasp ενώ άλλα παρέχουν τη δυνατότητα στο μαθητή να περιγράψει τα βήματα του αλγορίθμου οπτικά με τη χρήση κάποιας γραφικής διεπαφής (MatrixPro, Animal).

Ο βαθμός διαδραστικότητας και η επιλογή κατάλληλων αναπαραστάσεων για τις αλγοριθμικές δομές είναι τα δύο πιο σημαντικά χαρακτηριστικά, τα οποία καθορίζουν την αποτελεσματικότητα ενός συστήματος οπτικοποίησης αλγορίθμων. Ο κατάλληλος σχεδιασμός ενός τέτοιου συστήματος θα πρέπει να έχει ως στόχο την ανάδειξη των σημαντικών στοιχείων (γεγονότων) κάθε αλγορίθμου και την παροχή αυξημένων δυνατοτήτων αλληλεπίδρασης του μαθητή με το περιβάλλον της οπτικοποίησης.

Επίσης, η εξέλιξη του παγκόσμιου ιστού έχει κάνει επιτακτική την ανάγκη τα νέα περιβάλλοντα να μπορούν να είναι προσβάσιμα μέσω δυναμικών ιστοσελίδων οι οποίες εκτελούν κώδικα συνήθως στον υπολογιστή του χρήστη (client based). Τα συστήματα αυτά έχουν το πλεονέκτημα ότι είναι προσβάσιμα από οποιαδήποτε υπολογιστική πλατφόρμα η οποία παρέχει πρόσβαση στο διαδίκτυο όπως ταμπλέτες, κινητά τηλέφωνα κλπ. Επίσης οι εφαρμογές αυτές εκτελούνται σε όλα τα λειτουργικά συστήματα χωρίς να ο χρήστης να χρειαστεί να μεταφορτώσει και να εγκαταστήσει κάποιο λογισμικό όπως συμβαίνει με τις εφαρμογές στη γλώσσα Java.

## *3.5 Σκοπός και Ερευνητικά Ερωτήματα της Διατριβής*

Η παρούσα διατριβή έχει τους εξής στόχους :

**1.** Διερεύνηση των δυσκολιών και των παρανοήσεων που έχουν οι μαθητές στη δομή δεδομένων του πίνακα και στους αλγόριθμους ταξινόμησης.

**2.** Σχεδιασμός και ανάπτυξη ενός διαδικτυακού περιβάλλοντος οπτικοποίησης αλγορίθμων επεξεργασίας πινάκων.

**3.** Μελέτη της συμβολής του περιβάλλοντος οπτικοποίησης στην κατανόηση της λειτουργίας των αλγορίθμων ταξινόμησης από μαθητές Γ' Λυκείου.

Τα ερευνητικά ερωτήματα που τέθηκαν ήταν τα εξής:

- Ποιες είναι οι αναπαραστάσεις και οι παρανοήσεις μαθητών Γ' Λυκείου για την έννοια του πίνακα;
- Σε ποιο βαθμό οι μαθητές έχουν αναπτύξει ικανότητες επίλυσης αλγοριθμικών προβλημάτων που περιλαμβάνουν τη χρήση πινάκων; Μπορούν να διακρίνουν σε ποια προβλήματα είναι απαραίτητη η χρήση πίνακα και σε ποια όχι;
- Ποιες δυσκολίες εμφανίζουν οι μαθητές Γ' Λυκείου κατά την επίλυση προβλημάτων που απαιτούν τη χρήση αλγόριθμων ταξινόμησης;
- Σε ποιο βαθμό το διαδικτυακό περιβάλλον οπτικοποίησης DAVE ενισχύει τη γνωστική εμπλοκή των μαθητών κατά την επίλυση προβλημάτων με αλγόριθμους ταξινόμησης;

# 4

# *Μελέτη των προϋπαρχουσών αντιλήψεων μαθητών για τους αλγόριθμους ταξινόμησης*

## *4.1 Εισαγωγή*

Σύμφωνα με την θεωρία του εποικοδομισμού η νέα γνώση χτίζεται πάνω σε προϋπάρχουσα γνώση και στις εμπειρίες που έχουν οι μαθητές από την καθημερινότητά τους. Για αυτό η μελέτη της προϋπάρχουσας γνώσης των μαθητών για μια έννοια όπως οι αλγόριθμοι ταξινόμησης έχει πολύ μεγάλη σημασία για το σχεδιασμό των διδακτικών στρατηγικών από τους εκπαιδευτικούς. Οι αναπαραστάσεις που έχουν οι μαθητές για τους αλγόριθμους ταξινόμησης μπορούν επίσης να αξιοποιηθούν για τον σχεδιασμό οπτικοποιήσεων αλγορίθμων ταξινόμησης.

Στη βιβλιογραφία βρέθηκαν μόνο δύο έρευνες σχετικά με τις προϋπάρχουσες γνώσεις των μαθητών σε αλγόριθμους ταξινόμησης (Chen et al., 2007· Simon et. al., 2006). Οι ερευνητές αυτής της ομάδας μελετούν τις αντιλήψεις των μαθητών σε προχωρημένες έννοιες της πληροφορικής, όπως πολυπλοκότητα, λογική και άλλες. Οι ερευνητές ζήτησαν από τους φοιτητές να περιγράψουν τον τρόπο με τον οποίο θα ταξινομούσαν μια σειρά από 10 αριθμούς σε αύξουσα σειρά. Τα αποτελέσματα έδειξαν ότι οι περισσότεροι φοιτητές αντιμετώπισαν τους αριθμούς ως συμβολοσειρές από ψηφία και έκαναν συγκρίσεις ψηφίο-ψηφίο, ομαδοποιώντας τους αριθμούς σε δεκάδες, εκατοντάδες κλπ., χτίζοντας ένα ευρετήριο με βάση το πλήθος των ψηφίων. Η δεύτερη πιο δημοφιλής μέθοδος ήταν ο αλγόριθμος επιλογής (selection sort). Άλλοι αλγόριθμοι που χρησιμοποιήθηκαν από μαθητές ήταν ο αλγόριθμος εισαγωγής (insertion sort) και σε πολύ λίγες περιπτώσεις ο αλγόριθμος ευθείας ανταλλαγής (bubble sort). Επίσης το

30% των φοιτητών δεν μπόρεσαν να περιγράψουν σωστά έναν αλγόριθμο ταξινόμησης ενώ ενδιαφέρον παρουσιάζει το γεγονός ότι οι φοιτητές που είχαν διδαχθεί προγραμματισμό (με τη γλώσσα Java) τα πήγαν χειρότερα από αυτούς που δεν είχαν καμία εμπειρία ή εκπαίδευση στον προγραμματισμό.

Μια βασική διαφορά της δικής μας έρευνας από τις παραπάνω είναι ο έμμεσος ορισμός του υπολογιστικού μοντέλου μέσα από την εκφώνηση του προβλήματος που τέθηκε στους μαθητές, σε αντίθεση με τις άλλες έρευνες όπου δόθηκε στους φοιτητές μεγάλη ελευθερία έκφρασης στη διατύπωση του αλγορίθμου.

Αποτέλεσμα αυτής της ελευθερίας που δόθηκε στους φοιτητές ήταν, πολλές από τις απαντήσεις να μην "δείχνουν" κάποιον γνωστό αλγόριθμο ταξινόμησης, ή κάποιον αλγόριθμο ταξινόμησης που να είναι αποδεκτός από το υπολογιστικό μοντέλο που χρησιμοποιούμε. Για παράδειγμα ένας φοιτητής μπορεί να απαντήσει ότι με μια ματιά βρίσκει αμέσως το μικρότερο στοιχείο, χωρίς να περιγράψει αναλυτικά τα βήματα που ακολούθησε για να το πετύχει.

Το σύνολο των επιτρεπτών εντολών που μπορούν να χρησιμοποιηθούν ορίζει τη νοητή μηχανή που προγραμματίζουμε για την επίλυση του προβλήματος. Η πρώτη αναφορά στην έννοια της νοητής μηχανής (notional machine) έγινε από τον du Boulay(1986). Σύμφωνα με τον du Boulay η νοητή μηχανή περικλείει το σύνολο των γενικών χαρακτηριστικών που έχει η μηχανή στην οποία προγραμματίζουμε τη λύση ενός προβλήματος. Στη δική μας έρευνα η νοητή μηχανή την οποία προγραμματίζουν οι μαθητές προκύπτει έμμεσα από την εκφώνηση του προβλήματος που τους δόθηκε:

> *Σας δίνουν 20 κλειστούς φακέλους. Σε κάθε έναν από αυτούς υπάρχει ένα χαρτί με έναν αριθμό γραμμένο πάνω σε αυτό. Σας ζητείται να βάλετε τους φακέλους σε αύξουσα σειρά (δηλαδή από το μικρότερο στο μεγαλύτερο) με βάση τα νούμερα αυτά.*
>
> *Μπορείτε να ανοίξετε κάθε φάκελο όσες φορές θέλετε αλλά μπορείτε να έχετε ανοιχτούς την ίδια στιγμή το πολύ δύο φακέλους και να βλέπετε τι έχουν μέσα. Μετά τους κλείνετε πάλι και τους τοποθετείτε όπου εσείς κρίνετε.*
>
> *Να περιγράψετε όσο πιο αναλυτικά μπορείτε τον αλγόριθμο με τον οποίο θα βάλετε τους φακέλους στη σειρά.*

Ορίζουμε μια νοητή μηχανή στην οποία οι φάκελοι παίζουν το ρόλο των μεταβλητών και οι αριθμοί το περιεχόμενό τους. Το άνοιγμα ενός φακέλου μοντελοποιεί την προσπέλαση στη μνήμη της αντίστοιχης μεταβλητής. Επειδή κάθε φορά μπορεί να συγκριθούν μόνο δύο στοιχεία, θέσαμε τον περιορισμό των δύο ανοικτών φακέλων για κάθε βήμα του αλγορίθμου, ώστε να υποχρεώσουμε τους μαθητές να συγκρίνουν μόνο δύο αντικείμενα κάθε φορά.

Οι βασικοί κανόνες-περιορισμοί που θέσαμε έμμεσα στους μαθητές είναι οι εξής:

- Την ίδια στιγμή μπορούν να έχουν ανοικτούς το πολύ δύο φακέλους
- Αφού δουν το περιεχόμενο ενός φακέλου τον ξανακλείνουν και τον τοποθετούν πάλι στη θέση του, εκτός αν θέλουν να τον μετακινήσουν σε άλλο σημείο.

Επίσης, επιτρέψαμε στους μαθητές να τοποθετούν τους φακέλους όπου ήθελαν στο τραπέζι. Αυτή η απόφαση αποδείχθηκε κρίσιμη διότι όπως θα δούμε στην ανάλυση των αποτελεσμάτων, πολλοί μαθητές επινόησαν αναδρομικούς αλγόριθμους όπως η Merge Sort και η Quicksort οι οποίοι χρειάζονται πολλούς πίνακες. Η υλοποίηση πάνω στον ίδιο πίνακα (in-place) είναι αρκετά δύσκολη για αρχάριους προγραμματιστές και δεν προσφέρει κάτι σημαντικό στην κατανόηση της λειτουργίας του αλγόριθμου. Αποτελεί πρακτική για τη βέλτιστη χρήση της μνήμης. Επίσης οι in-place αλγόριθμοι δεν είναι αλγόριθμοι που χρησιμοποιούνται στην καθημερινή ζωή συχνά, διότι στον φυσικό κόσμο δεν έχουμε τους περιορισμούς στη μνήμη που μας επιβάλλει η αρχιτεκτονική των υπολογιστών. Είναι συνηθισμένο όταν βάζουμε στη σειρά μια ομάδα από αντικείμενα να τοποθετούμε τα ταξινομημένα ξεχωριστά από τα υπόλοιπα και όχι δίπλα τους. Αυτό το είδαμε και στην παρούσα έρευνα. Η τοποθέτηση των ταξινομημένων αντικειμένων σε ξεχωριστή περιοχή υποδηλώνει τη χρήση μιας επιπλέον δομής δεδομένων, αν προσπαθήσουμε να μεταφράσουμε το πρόγραμμα της νοητής μηχανής σε μια συμβατική γλώσσα προγραμματισμού.

## *4.2 Μεθοδολογία της Έρευνας*

Η έρευνα διεξήχθη σε δύο πειραματικά σχολεία στο Ζάννειο Πειραματικό Γυμνάσιο και στο Ζάννειο Πειραματικό Λύκειο. Στην πρώτη περίπτωση το δείγμα ήταν 180 μαθητές των δύο πρώτων τάξεων του Γυμνασίου και στη δεύτερη 106 μαθητές όλων των τάξεων του Λυκείου. Όλοι οι μαθητές είχαν διδαχθεί κάποιες βασικές έννοιες προγραμματισμού σε περιβάλλον οπτικού προγραμματισμού με πλακίδια (Scratch) και

στη γλώσσα Logo. Δεν είχαν όμως ασχοληθεί καθόλου με δομές δεδομένων και δεν γνώριζαν κανέναν αλγόριθμο ταξινόμησης. Μόνο τέσσερις (4) μαθητές της τεχνολογικής κατεύθυνσης της Γ' Λυκείου είχαν διδαχθεί τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής (φυσαλίδας) στο πλαίσιο του μαθήματος της Ανάπτυξης Εφαρμογών σε Προγραμματιστικό Περιβάλλον.

Στη συνέχεια δίνουμε την κατανομή των μαθητών ανά σχολείο και τάξη.

**Πίνακας 4.1. Κατανομή των μαθητών του δείγματος**

| **Τάξη** | **Μαθητές** | **Κατεύθυνση** | **Μαθητές** |
|---|---|---|---|
| Α' Γυμνασίου | 80 | | |
| Β' Γυμνασίου | 100 | | |
| Α' Λυκείου | 39 | | |
| Β' Λυκείου | 35 | | |
| Γ' Λυκείου | 32 | Θεωρητική | 15 |
| | | Θετική | 10 |
| | | Τεχνολογική 1 | 4 |
| | | Τεχνολογική 2 | 3 |
| Σύνολο | 286 | | |

Για τη συμπλήρωση του ερωτηματολογίου από τους μαθητές διατέθηκε μια διδακτική ώρα. Το ερωτηματολόγιο ήταν επώνυμο, ώστε να έχουμε στη συνέχεια τη δυνατότητα να διερευνήσουμε ακόμη περισσότερο, μέσω συνεντεύξεων, τις ιδέες των μαθητών που θα παρουσίαζαν ερευνητικό ενδιαφέρον. Περιλάμβανε το πρόβλημα με τους φακέλους που αναλύθηκε παραπάνω. Ζητήθηκε από τους μαθητές να επινοήσουν και να διατυπώσουν έναν αλγόριθμο ταξινόμησης σε όποια μορφή ήθελαν και με όσο πιο αναλυτικό τρόπο μπορούσαν.

Τα βασικά ερευνητικά ερωτήματα αυτής της έρευνας είναι τα παρακάτω:

- Με ποια μέθοδο / αλγόριθμο ταξινομούν οι μαθητές μια σειρά από αντικείμενα;
- Πως περιγράφουν οι μαθητές τα βήματα ενός αλγορίθμου;
- Πως περιγράφουν/κωδικοποιούν μια επαναληπτική διαδικασία;

- Διαφέρουν οι αναπαραστάσεις των μαθητών Γυμνασίου για τη διαδικασία της ταξινόμησης από αυτές των μαθητών Λυκείου;

## *4.3 Αποτελέσματα της έρευνας*

Στον Πίνακα 4.2 παρουσιάζονται τα συγκεντρωτικά αποτελέσματα της έρευνας για τους μαθητές Γυμνασίου. Οι περισσότεροι μαθητές (53.4%) χρησιμοποίησαν τους αλγόριθμους επιλογής και εισαγωγής. Τα αποτελέσματα αυτά συμφωνούν σε κάποιο βαθμό με τις έρευνες των (Chen et. al., 2007· Simon et al., 2006). Ωστόσο υπάρχουν και σημαντικές διαφορές που θα αναλύσουμε στη συνέχεια και οφείλονται στη μεθοδολογία που ακολουθήσαμε.

**Πίνακας 4.2. Απαντήσεις των μαθητών Γυμνασίου ανά τάξη**

| | | **Τάξη** | | **Σύνολο** | |
|---|---|---|---|---|---|
| **α/α** | **Απάντηση (N=180)** | **A** | **B** | **Αριθμός** | **Ποσοστό (%)** |
| 1 | Δεν απάντησαν | 1 | 7 | 8 | 4.4 |
| 2 | Ημιτελής περιγραφή | 30 | 35 | 65 | 36.1 |
| 3 | Ταξινόμηση Εισαγωγής | 22 | 26 | 48 | 26.7 |
| 4 | Ταξινόμηση Επιλογής | 25 | 23 | 48 | 26.7 |
| 5 | Ταξινόμηση Συγχώνευσης | 2 | 8 | 10 | 5.6 |
| 6 | Ταξινόμηση Φυσαλίδας | 0 | 1 | 1 | 0.5 |

Εδώ πρέπει να σημειώσουμε ότι οι περισσότερες απαντήσεις των μαθητών εξετάστηκαν με αρκετή επιείκεια όσον αφορά την σαφήνεια και την αυστηρότητα διατύπωσης των αλγόριθμων που πρότειναν. Αυτό έγινε επειδή μας ενδιέφερε κυρίως να διακρίνουμε το είδος του αλγορίθμου από τη βασική ιδέα και όχι να σταθούμε σε συντακτικά ή εκφραστικά λάθη. Για αυτό είμασταν αρκετά ελαστικοί όσον αφορά την παραβίαση των κριτηρίων της αποτελεσματικότητας και της καθοριστικότητας των αλγόριθμων που έδωσαν οι μαθητές.

Στον Πίνακα 4.3 δίνονται τα συγκεντρωτικά αποτελέσματα για τους μαθητές Λυκείου:

**Πίνακας 4.3. Απαντήσεις των μαθητών Λυκείου ανά τάξη**

| α/α | Απάντηση (N=106) | Τάξη | | | Σύνολο | |
|---|---|---|---|---|---|---|
| | | Α | Β | Γ | Αριθμός | Ποσοστό (%) |
| 1 | Δεν απάντησαν | 2 | 0 | 0 | 2 | 1.9 |
| 2 | Ημιτελής περιγραφή | 1 | 10 | 0 | 11 | 10.4 |
| 3 | Ταξινόμηση Εισαγωγής | 23 | 10 | 20 | 53 | 50 |
| 4 | Ταξινόμηση Επιλογής | 4 | 7 | 5 | 16 | 15 |
| 5 | Ταξινόμηση Συγχώνευσης | 6 | 6 | 3 | 15 | 14.2 |
| 6 | Ταξινόμηση Φυσαλίδας | 0 | 0 | 4 | 4 | 3.8 |
| 7 | Γρήγορη Ταξινόμηση | 3 | 2 | 0 | 5 | 4.7 |

Το 55% των μαθητών επινόησε τον αλγόριθμο ταξινόμησης εισαγωγής, το 18% τον αλγόριθμο επιλογής και το 12% τον αλγόριθμο συγχώνευσης. Τον αλγόριθμο της φυσαλίδας τον χρησιμοποίησαν μόλις 4 μαθητές από τους οποίους οι 3 ήταν μαθητές της τεχνολογικής κατεύθυνσης. Εκτιμούμε ότι τον είχαν διδαχθεί και αυτό φάνηκε από το γεγονός ότι έδωσαν αλγόριθμο με τη μορφή ψευδογλώσσας.

Στη συνέχεια δίνουμε κάποιες χαρακτηριστικές απαντήσεις μαθητών για κάθε περίπτωση. Να σημειωθεί ότι στις απαντήσεις των μαθητών έχουμε παρέμβει μόνο ως προς τη διόρθωση των ορθογραφικών λαθών που υπήρχαν.

### *4.3.1 Ταξινόμηση Εισαγωγής (Insertion sort)*

Ο αλγόριθμος ταξινόμησης με εισαγωγή είναι ο ένας από τους δύο αλγόριθμους που χρησιμοποίησαν οι περισσότεροι μαθητές Γυμνασίου. Μια χαρακτηριστική συνοπτική περιγραφή μαθητή γυμνασίου είναι η παρακάτω:

*«Χωρίζω τους φακέλους σε δύο ομάδες, αυτούς που είναι ταξινομημένοι και αυτούς που δεν είναι. Κάθε φορά παίρνω έναν από τους μη ταξινομημένος και τον βάζω στην σωστή θέση στους ταξινομημένους.»*

Στην παραπάνω απάντηση φαίνεται ξεκάθαρα η τάση που έχουν οι μαθητές να ομαδοποιούν τα αντικείμενα κατά την ταξινόμηση σε δύο σύνολα, τα ταξινομημένα

και τα μη ταξινομημένα. Διαδοχικά αφαιρούν ένα στοιχείο από το αρχικό σύνολο και το εισάγουν στο ήδη ταξινομημένο σύνολο αφού πρώτα βρουν τη θέση του.

Μια άλλη χαρακτηριστική απάντηση είναι η παρακάτω:

*«Ανοίγω τον πρώτο και τον δεύτερο φάκελο. Τους συγκρίνω, βάζω τον μικρότερο αριστερά και τον μεγαλύτερο δεξιά. Ανοίγω τον επόμενο φάκελο και τον συγκρίνω με τον μικρότερο που έβαλα στην άκρη. Αν είναι μικρότερος από αυτόν τον βάζω αριστερά του ενώ αν είναι μεγαλύτερος δεξιά του. Ανοίγω τον επόμενο φάκελο, αν είναι μικρότερος από τον μικρότερο τον βάζω πρώτο ενώ αν είναι μεγαλύτερος από τον μεγαλύτερο τελευταίο, αλλιώς τον βάζω στην μέση. Συνεχίζω με τους υπόλοιπους φακέλους.»*

Η δομή αυτής της απάντησης είναι και η πιο συνηθισμένη που είδαμε. Δηλαδή οι μαθητές περιγράφουν τα 2-3 πρώτα βήματα και τελειώνουν με μια φράση όπως "κ.ο.κ", "ομοίως" κλπ. Δεν δίνουν δηλαδή τον αλγόριθμο στην μορφή που γνωρίζουμε χρησιμοποιώντας τις γνωστές δομές ακολουθίας, επιλογής και επανάληψης, αλλά με ελεύθερο κείμενο στο οποίο ωστόσο είναι διακριτή η βασική ιδέα του αλγορίθμου.

Επίσης είναι φανερό ότι η μαθήτρια που έδωσε την παραπάνω απάντηση έχει χωρίσει τους φακέλους σε δύο ομάδες, τους ταξινομημένους και τους μη ταξινομημένους. Κάθε φορά παίρνει έναν φάκελο από το αρχικό σύνολο και το τοποθετεί στη σωστή θέση μεταξύ των ταξινομημένων φακέλων, μέχρι να τους τοποθετήσει όλους. Δεν το αναφέρει ρητά γιατί το θεωρεί αυτονόητο. Στη συνέντευξη όμως που ακολούθησε παραδέχτηκε ότι ακολούθησε αυτή την προσέγγιση.

Ο αλγόριθμος ταξινόμησης με εισαγωγή ήταν πολύ περισσότερο δημοφιλής μεταξύ των μαθητών λυκείου αφού τον χρησιμοποίησαν περισσότεροι από τους μισούς. Παραθέτουμε και σχολιάζουμε κάποιες χαρακτηριστικές απαντήσεις μαθητών λυκείου:

*«Παίρνω τους δύο πρώτους και τους βάζω στη σειρά. Στη συνέχεια παίρνω τον 3$^{ο}$ και τον τοποθετώ στη σωστή θέση (πρώτο, τελευταίο ή ανάμεσα στους 2). Κάνω το ίδιο για τους υπόλοιπους αριθμούς τοποθετώντας κάθε φορά τον επόμενο στην σωστή θέση.»*

Η δομή αυτής της απάντησης είναι και η πιο συνηθισμένη που είδαμε. Δηλαδή οι μαθητές περιγράφουν τα 2-3 πρώτα βήματα και τελειώνουν με μια φράση όπως "κ.ο.κ", "ομοίως" κλπ. Δεν δίνουν δηλαδή τον αλγόριθμο στην μορφή που γνωρίζουμε χρησιμοποιώντας τις γνωστές δομές ακολουθίας, επιλογής και επανάληψης, αλλά με ελεύθερο κείμενο στο οποίο ωστόσο είναι διακριτή η βασική ιδέα του αλγορίθμου.

Επίσης είναι φανερό ότι η μαθήτρια που έδωσε την παραπάνω απάντηση έχει χωρίσει τους φακέλους σε δύο ομάδες, τους ταξινομημένους και τους μη ταξινομημένους. Κάθε φορά παίρνει έναν φάκελο από το αρχικό σύνολο και το τοποθετεί στη σωστή θέση μεταξύ των ταξινομημένων φακέλων, μέχρι να τους τοποθετήσει όλους. Δεν το αναφέρει ρητά γιατί μάλλον το θεωρεί αυτονόητο. Αυτό είναι πιο ξεκάθαρο στην παρακάτω απάντηση μαθητή:

*«Χωρίζω τους φακέλους σε δύο ομάδες, αυτούς που είναι ταξινομημένοι και αυτούς που δεν είναι. Κάθε φορά παίρνω έναν από το πάνω μέρος της στοίβας των μην ταξινομημένων και τον βάζω στην σωστή θέση στους ταξινομημένους.»*

Στην παραπάνω απάντηση φαίνεται ξεκάθαρα η τάση που έχουν οι μαθητές να ομαδοποιούν τα αντικείμενα κατά την ταξινόμηση σε δύο σύνολα, τα ταξινομημένα και τα μη ταξινομημένα. Βαθμιαία αφαιρούν ένα στοιχείο από το αρχικό σύνολο και το εισάγουν στο ήδη ταξινομημένο σύνολο αφού πρώτα βρουν τη θέση του.

*«Βάζω τον πρώτο φάκελο στο κέντρο. Αν ο επόμενος είναι μεγαλύτερος τον βάζω δεξιά και αν είναι μικρότερος αριστερά. Αφήνω αρκετή απόσταση ώστε να χωρέσουν και άλλοι φάκελοι αν υπάρχουν μεταξύ αυτών που έχω ήδη βάλει. Επαναλαμβάνω τη διαδικασία μέχρι να τελειώσουν οι φάκελοι.»*

Η παραπάνω προσέγγιση αποτελεί ένα ακόμα παράδειγμα του περιορισμού της εφευρετικότητας και της φαντασίας των μαθητών που επιτελείται από τη διδασκαλία του προγραμματισμού μέσα σε συγκεκριμένο πλαίσιο με αυστηρούς κανόνες. Οι μαθητές που δεν είχαν διδαχθεί προγραμματισμό παρουσίασαν πρωτότυπες λύσεις οι οποίες δεν περιορίζονται από τους κανόνες που θέτει μια εκπαιδευτική γλώσσα προγραμματισμού. Η ικανότητα των μαθητών να ανακαλύπτουν εκ νέου τον τροχό ήταν εντυπωσιακή.

Ο παραπάνω αλγόριθμος ενώ παραπέμπει στην βασική ιδέα του αλγόριθμου εισαγωγής δεν ταυτίζεται με αυτόν. Θα μπορούσαμε να πούμε ότι είναι μια παραλλαγή του η οποία είναι σχεδιασμένη για ένα φυσικό νοητό μοντέλο υπολογισμού, το οποίο περιλαμβάνει τις κινήσεις που μπορεί να κάνει κάποιος σε ένα τραπέζι με απεριόριστη επιφάνεια. Η μεταφορά αυτού του αλγόριθμου σε μια γλώσσα προγραμματισμού δεν είναι τόσο απλή υπόθεση, ειδικά αν δεν έχουμε αρκετή μνήμη.

### *4.3.2 Ταξινόμηση Επιλογής (Selection sort)*

Ο άλλος δημοφιλής αλγόριθμος που επινοήθηκε από τους μαθητές γυμνασίου για την επίλυση του προβλήματος ήταν ο αλγόριθμος με επιλογή (selection sort). Ενώ

στους μαθητές Λυκείου ο αλγόριθμος αυτός ήταν δεύτερος στις επιλογές τους με μεγάλη διαφορά από τον πρώτο που ήταν ο αλγόριθμος με εισαγωγή, στους μαθητές Γυμνασίου χρησιμοποιήθηκε όσες και ο αλγόριθμος με εισαγωγή. Παρακάτω δίνουμε μερικές χαρακτηριστικές απαντήσεις μαθητών γυμνασίου:

*«Αρχικά θα πάρουμε 2 φακέλους και να τους συγκρίνουμε, τον μικρότερο τον κρατάμε, ενώ τον άλλον τον κλείνουμε. Αφού λοιπόν κάνουμε αυτή τη διαδικασία σε όλους τους φακέλους, βάζουμε τον μικρότερο στην αρχή. Με την ίδια διαδικασία βρίσκουμε τον αμέσως μικρότερο και τον βάζουμε δεύτερο. Αυτό θα γίνει και στους υπόλοιπους φακέλους με βάση την προηγούμενη διαδικασία.»*

Από την παραπάνω περιγραφή φαίνεται η απλότητα του αλγόριθμου με επιλογή αφού συνίσταται σε μια επαναλαμβανόμενη εύρεση μεγίστου κάτι που επίσης καθιστά την υλοποίησή του αρκετά απλή.

Η παραπάνω απάντηση περιγράφει αρκετά καλά τον αλγόριθμο επιλογής. Ωστόσο παρατηρούμε ότι και εδώ οι μαθητές αποθηκεύουν τους ταξινομημένους αριθμούς σε ξεχωριστή δομή. Αν για παράδειγμα υλοποιήσουμε τον παραπάνω αλγόριθμο με πίνακες θα χρειαστούμε δύο πίνακες. Έναν για τα αρχικά δεδομένα και έναν για τους ταξινομημένους αριθμούς. Οι μαθητές έδωσαν και μερικές παραλλαγές της ταξινόμησης επιλογής. Σε μια από αυτές υπολογίζουν σε κάθε βήμα και το μέγιστο και το ελάχιστο στοιχείο του πίνακα και τοποθετούν το ένα στην αρχή και το άλλο στο τέλος του νέου πίνακα.

Εδώ θα πρέπει να σημειώσουμε ότι ενώ αναμέναμε ότι αυτός θα ήταν ο δημοφιλέστερος αλγόριθμος, οι μαθητές Λυκείου μας διέψευσαν και προτίμησαν τον αλγόριθμο εισαγωγής. Η υπόθεσή μας βασιζόταν στο γεγονός ότι ο αλγόριθμος επιλογής είναι ο απλούστερος αλγόριθμος αφού αφορά μια επαναλαμβανόμενη εύρεση μεγίστου κάτι που καθιστά την υλοποίησή του αρκετά απλή και κατανοητή. Αυτό όμως ισχύει για το υπολογιστικό μοντέλο του υπολογιστή το οποίο διαφέρει από τη νοητή μηχανή που περιγράφει τις φυσικές κινήσεις που μπορούν να κάνουν οι μαθητές με τα χέρια τους. Παρακάτω δίνεται μια χαρακτηριστική απάντηση μαθητή Λυκείου:

*«Ανοίγουμε 2 φακέλους και κρατάμε τον φάκελο με τον μεγαλύτερο αριθμό, ενώ τον άλλο τον βάζουμε πίσω στην στοίβα με τους άλλους που έχουμε ανοίξει. Το κάνουμε αυτό μέχρι να ανοιχτούν όλοι οι φάκελοι και να βρεθούν στη στοίβα. Αυτός που μένει στο χέρι μας είναι ο μεγαλύτερος. Επαναλαμβάνουμε την ίδια διαδικασία και ο φάκελος που μένει στο χέρι μας είναι ο δεύτερος μεγαλύτερος. Τον τοποθετούμε δίπλα στον πρώτο φάκελο. Επαναλαμβάνουμε άλλες 18 φορές. Στο τέλος όλοι οι φάκελοι θα είναι στη σειρά.»*

Η παραπάνω απάντηση περιγράφει αρκετά καλά τον αλγόριθμο επιλογής. Ωστόσο παρατηρούμε ότι και εδώ οι μαθητές αποθηκεύουν τους ταξινομημένους αριθμούς σε ξεχωριστή δομή. Αν για παράδειγμα υλοποιήσουμε τον παραπάνω αλγόριθμο με πίνακες θα χρειαστούμε δύο πίνακες. Έναν για τα αρχικά δεδομένα και έναν για τους ταξινομημένους αριθμούς.

Αρκετοί μαθητές γυμνασίου και λυκείου έδωσαν και μια παραλλαγή της ταξινόμησης επιλογής, στην οποία υπολογίζουν σε κάθε βήμα παράλληλα το μέγιστο και το ελάχιστο στοιχείο του πίνακα και τοποθετούν το ένα στην αρχή και το άλλο στο τέλος του νέου πίνακα.

### *4.3.3 Ταξινόμηση Συγχώνευσης (Merge sort)*

Οι περιγραφές των μαθητών που πρότειναν αυτόν τον αλγόριθμο δεν είναι ιδιαίτερα ακριβείς, κυρίως λόγω της δυσκολίας περιγραφής του αναδρομικού βήματος και της συγχώνευσης των ταξινομημένων λιστών. Ακολουθούν δύο ενδεικτικές απαντήσεις μαθητών Γυμνασίου:

*«Για να μπορέσω να βάλω και τους 20 φακέλους σε αύξουσα σειρά, θα ανοίξω τους πρώτους δύο και τους χωρίζω σε μικρό και μεγάλο και συνεχίζω να κάνω το ίδιο και με τους υπόλοιπους 18. Αφού, τους έχω ξεχωρίσει σε μικρούς και μεγάλους, ανοίγω δύο φακέλους από τους μικρότερους αριθμούς και τους χωρίζω σε μεγάλους και μικρούς. Το ίδιο κάνω με τους μεγαλύτερους. Μετά αρχίζω και τοποθετώ τους μικρότερους, πιο μετά τους μεγαλύτερους κ.ο.κ.»*

*«Εγώ θα πάρω 5 φακέλους θα τους ανοίξω έναν-έναν (και θα τους κλείνω φυσικά) και θα τους βάλω στη σειρά. Μετά θα πάρω άλλους 5 και θα τους βάλω στη σειρά. Θα φτιάξω 4 σειρές από 5 φακέλους. Μετά 2 σειρές των 10 φακέλων, μετά θα προσθέσω άλλους 5 και μετά άλλους 5.»*

Οι παραπάνω περιγραφές παραπέμπουν σε αλγόριθμο διαίρει και βασίλευε, αφού το πρόβλημα διαιρείται διαδοχικά σε απλούστερα προβλήματα. Οι μαθητές δυσκολεύονται πολύ στην περιγραφή της ένωσης των δύο ταξινομημένων λιστών. Ωστόσο φαίνεται ότι οι μαθητές έχουν κατανοήσει ότι η διαίρεση του προβλήματος οδηγεί σε άλλα απλούστερα.

Κάποιοι μαθητές διαχώρισαν τους αριθμούς με κριτήριο το πλήθος των ψηφίων (μονάδες, δεκάδες, εκατοντάδες). Αυτή την προσέγγιση ακολούθησαν οι περισσότεροι φοιτητές στις δύο διεθνείς έρευνες (Chen et al., 2007· Simon et al. 2006) της ομάδας "common sense computing".

*«Πρώτα θα χωρίσω τους αριθμούς σε μονοψήφιους, διψήφιους, τριψήφιους κλπ. Μετά ανά ομάδα τους βάζω σε αύξουσα σειρά και μετά ενώνω τις ομάδες, βάζοντας μπροστά τους μονοψήφιους, μετά τους διψήφιους, κλπ.»*

Τέλος θα πρέπει να σημειώσουμε ότι από τους 180 μαθητές μόνο ένας έδωσε αλγόριθμο ο οποίος μοιάζει με τον αλγόριθμο ευθείας ανταλλαγής (φυσαλίδας). Αυτό επιβεβαιώνει τα ευρήματα άλλων εργασιών (Βραχνός & Τζιμογιάννης, 2016· 2014α), σύμφωνα με τα οποία η ταξινόμηση φυσαλίδας δεν συνάδει με τις αρχές του εποικοδομισμού, αφού είναι πολύ δύσκολο όχι μόνο να χτιστεί σε προϋπάρχουσα γνώση των μαθητών αλλά και να συνδεθεί με εμπειρίες τους από την καθημερινή ζωή. Αυτός ίσως να είναι ο βασικότερος λόγος που οι μαθητές δυσκολεύονται τόσο πολύ στην κατανόηση της λειτουργίας του.

Παρόμοιες απαντήσεις καταγράφηκαν και από τους μαθητές Λυκείου:

*«Φτιάχνω 20 θέσεις στο χώρο τις οποίες χωρίζω στα δύο, δέκα δεξιά και δέκα αριστερά. Στη συνέχεια χωρίζω τους 10 πάλι σε ομάδες των 5 και τις πεντάδες σε δυάδες και τριάδες. Βάζω τις δυάδες και τις τριάδες στη σειρά και στη συνέχεια ενώνω αντίστροφα μέχρι να φτάσω στις δύο δεκάδες οπότε ενώνω και αυτές.»*

Η ιδέα μοιάζει αρκετά με αυτή της συγχώνευσης. Το μόνο πρόβλημα είναι η έλλειψη περιγραφής του βήματος της ένωσης των δύο ταξινομημένων λιστών. Αυτό που έχει σημασία είναι ότι ο μαθητής δείχνει να έχει κατανοήσει πολύ καλά ότι η διαίρεση του προβλήματος οδηγεί σε απλούστερα προβλήματα.

### *4.3.4 Γρήγορη Ταξινόμηση (Quick Sort)*

Συνολικά πέντε μαθητές λυκείου (τρεις από την Α' τάξη) έδωσαν απαντήσεις οι οποίες έχουν αρκετά κοινά στοιχεία με τον δημοφιλή αλγόριθμο της γρήγορης ταξινόμησης (QuickSort), με πιο χαρακτηριστικό τον διαχωρισμό των αριθμών σε δύο λίστες με βάση την τιμή ενός στοιχείου του πίνακα. Οι αναδρομικές κλήσεις είναι πολύ δύσκολο να περιγραφούν σωστά από τους μαθητές και για αυτό είδαμε απαντήσεις όπως "κάνω το ίδιο και για τα δύο σύνολα", "επιλέγω πάλι έναν αριθμό κ.ο.κ.". "επαναλαμβάνω την ίδια διαδικασία" κλπ. Δηλαδή οι μαθητές προσπαθούν να περιγράψουν τα αναδρομικά βήματα με παρόμοιο τρόπο με τις επαναληπτικές διαδικασίες. Αυτό αποδεικνύει ότι οι μαθητές αντιλαμβάνονται ότι η αναδρομή συνδέεται με την έννοια της επαναληπτικής διαδικασίας (Rinderknecht, 2014).

Ακολουθεί μια περιγραφή της Quicksort από μαθητή της Α' Λυκείου.

*«Παίρνω στην τύχη έναν φάκελο και τον βάζω στη μέση. Μετά ανοίγω όλους τους φακέλους έναν-έναν και αν είναι μικρότεροι τους βάζω αριστερά ενώ αν είναι μεγαλύτεροι δεξιά. Στη συνέχεια από τους μικρότερους ανοίγω πάλι έναν στην τύχη και επαναλαμβάνω την ίδια διαδικασία. Το ίδιο και για τους μεγαλύτερους.»*

### *4.3.5 Ταξινόμηση Ευθείας Ανταλλαγής/Φυσαλίδας (Bubble Sort)*

Ο αλγόριθμος ταξινόμησης φυσαλίδας προτάθηκε μόνο από 4 μαθητές της Γ' τάξης του Λυκείου και έναν μαθητή της Β' Γυμνασίου. Ωστόσο οι 3 μαθητές τον είχαν ήδη διδαχθεί στο μάθημα της ανάπτυξης εφαρμογών της τεχνολογικής κατεύθυνσης. Ουσιαστικά μόνο 2 στους 286 μαθητές επινόησαν αυτόν τον αλγόριθμο ταξινόμησης. Οι αλγόριθμοι των τριών μαθητών ήταν γραμμένοι στην ψευδογλώσσα του σχολικού βιβλίου. Παρακάτω δίνουμε την απάντηση της μαθήτριας της θετικής κατεύθυνσης η οποία παρουσιάζει ενδιαφέρον ως προς τον τρόπο διατύπωσης.

*1. Τοποθετώ όλους τους φακέλους στη σειρά.*

*2. Ανοίγω τον πρώτο και τον δεύτερο φάκελο και τους συγκρίνω*

*3. Διαδικασία 1*

| |
|---|
| *Αν ο πρώτος είναι μικρότερος ή ίσος από τον δεύτερο τους αφήνω όπως ήταν* |
| *Αν ο δεύτερος είναι μικρότερος τους αλλάζω θέση* |

*4. Ανοίγω τον δεύτερο και τον τρίτο φάκελο*

*5. Επαναλαμβάνω τη διαδικασία 1 μέχρι τον τελευταίο φάκελο μέχρι οι αριθμοί να είναι στη σειρά*

Η περιγραφή της μαθήτριας μοιάζει αρκετά με φυσική γλώσσα κατά βήματα, αφού η παραπομπή στη Διαδικασία 1, δείχνει ουσιαστικά άλμα σε προηγούμενη εντολή.

Είναι φανερό λοιπόν ότι η ταξινόμηση φυσαλίδας δεν συνάδει με τις αρχές του εποικοδομισμού, αφού είναι πολύ δύσκολο όχι μόνο να χτιστεί σε προϋπάρχουσα γνώση των μαθητών αλλά και να συνδεθεί με εμπειρίες τους από την καθημερινή ζωή, αφού σχεδόν κανείς δεν πρόκειται να ταξινομήσει ένα σύνολο αντικειμένων με τα χέρια του χρησιμοποιώντας αυτόν τον αλγόριθμο. Αυτός ίσως να είναι ο βασικότερος λόγος που οι μαθητές δυσκολεύονται τόσο πολύ στην κατανόηση της λειτουργίας του (Βραχνός & Τζιμογιάννης, 2014α).

## *4.4 Συζήτηση*

Στον Πίνακα 4.4 παρουσιάζονται συγκριτικά τα συγκεντρωτικά αποτελέσματα των δύο ερευνών.

**Πίνακας 4.4. Απαντήσεις των μαθητών ανά τύπο σχολείου με ποσοστά %**

| α/α | Απάντηση | Γυμνάσιο (N=180) | Λύκειο (N=106) |
|---|---|---|---|
| 1 | Δεν απάντησαν | 4.4 | 1.9 |
| 2 | Ημιτελής περιγραφή αλγορίθμου | 36.1 | 10.4 |
| 3 | Ταξινόμηση Εισαγωγής | 26.7 | 50 |
| 4 | Ταξινόμηση Επιλογής | 26.7 | 15 |
| 5 | Ταξινόμηση Συγχώνευσης | 5.6 | 14.2 |
| 6 | Ταξινόμηση Φυσαλίδας | 0.5 | 3.8 |
| 7 | Γρήγορη Ταξινόμηση | 0 | 4.7 |

Οι μαθητές Λυκείου χρησιμοποίησαν τον αλγόριθμο εισαγωγής πολύ περισσότερο από αυτόν της επιλογής. Για κάποιο λόγο οι μαθητές Λυκείου έκαναν την υπόθεση ότι τα δεδομένα έρχονται σταδιακά και δεν τα έχουν όλα στη διάθεσή τους από την αρχή. Αυτή η σημαντική διαφορά στον τρόπο σκέψης ίσως να οφείλεται στην ωριμότητα που έχουν οι μαθητές Λυκείου στον προγραμματισμό μια και έχουν διδαχθεί κάποια έστω πολύ βασικά μαθήματα προγραμματισμού σε προηγούμενες τάξεις. Σίγουρα είναι κάτι που χρήζει περαιτέρω διερεύνησης.

Μια άλλη σημαντική διαφορά είναι ότι το 4.4% των μαθητών του Γυμνασίου δεν απάντησε καθόλου ενώ το 36% έδωσε απαντήσεις οι οποίες είχαν σοβαρές ελλείψεις ή λάθη και δεν μπορούσαν να κατηγοριοποιηθούν σε κάποιον γνωστό αλγόριθμο ταξινόμησης. Το ποσοστό αυτό (40%) είναι πολύ μεγάλο δεδομένου ότι το αντίστοιχο ποσοστό στο Λύκειο ήταν μόλις 12.4%. Είναι φανερό ότι οι μαθητές Γυμνασίου δυσκολεύτηκαν πολύ περισσότερο από τους μαθητές Λυκείου στη διατύπωση του αλγορίθμου κάτι που ήταν αναμενόμενο. Επίσης οι μαθητές Λυκείου είχαν παρακολουθήσει περισσότερα μαθήματα προγραμματισμού στη σχολική τους ζωή από ότι οι μαθητές γυμνασίου και έχουν σίγουρα μεγαλύτερη ευχέρεια στον γραπτό λόγο.

## *4.5 Συμπεράσματα*

Τα αποτελέσματα έδειξαν ότι οι μαθητές έχουν ήδη επινοήσει αλγόριθμους για την ταξινόμηση αντικειμένων σε αύξουσα ή φθίνουσα σειρά με βάση κάποιο κριτήριο, πριν τους διδαχθούν. Οι δύο πιο δημοφιλείς αλγόριθμοι που χρησιμοποιούν οι μαθητές είναι οι αλγόριθμοι εισαγωγής και επιλογής. Από αυτό μπορούμε να συμπεράνουμε ότι οι αλγόριθμοι αυτοί προκύπτουν από την κοινή λογική και έχουν σχέση με τις καθημερινές εμπειρίες των μαθητών. Για αυτό προτείνεται να είναι οι πρώτοι αλγόριθμοι ταξινόμησης με τους οποίους θα έρθουν σε επαφή οι αρχάριοι προγραμματιστές.

Ένα άλλο συμπέρασμα που προέκυψε από την έρευνα είναι ότι οι μαθητές προτιμούν να χρησιμοποιούν επιπλέον δομές δεδομένων για να διαχωρίζουν τα ταξινομημένα από τα μη ταξινομημένα αντικείμενα. Αυτό έρχεται σε αντίθεση με την κυριαρχούσα λογική των αλγορίθμων ταξινόμησης που εκτελούνται πάνω στον ίδιο πίνακα (in-place). Εκτιμούμε ότι αν οι εκπαιδευτικοί κατά την παρουσίαση των αλγορίθμων ταξινόμησης χρησιμοποιούσαν βοηθητικές δομές ώστε να φαίνεται ο διαχωρισμός ταξινομημένων/μη ταξινομημένων θα βοηθούσε πολύ τους μαθητές. Ο διαχωρισμός αυτός θα μπορούσε να φαίνεται σε κάποια οπτικοποίηση με ανάλογο χρωματισμό του κάθε τμήματος, ειδικά σε σύνθετους αλγορίθμους όπως αυτός της συγχώνευσης.

Ο αλγόριθμος εισαγωγής ήταν ο αλγόριθμος που είχε χρησιμοποιηθεί περισσότερο από μαθητές Λυκείου σε αντίστοιχη έρευνα. Μια σημαντική διαφορά που υπήρξε σε σχέση με την αντίστοιχη έρευνα σε μαθητές Λυκείου ήταν ότι πολλοί μαθητές Γυμνασίου έδωσαν ημιτελή ή μη ορθά τμήματα αλγορίθμων. Ένα πολύ μικρό ποσοστό μαθητών επινόησε τον αλγόριθμο συγχώνευσης επιλέγοντας διάφορα κριτήρια για τον διαχωρισμό των αριθμών, όπως συνέβη και στην περίπτωση των μαθητών του Λυκείου. Η σημαντικότερη διαφορά μεταξύ των δύο ερευνών ήταν ότι στην περίπτωση του Λυκείου ο αλγόριθμος της εισαγωγής ήταν ο επικρατέστερος με μεγάλη διαφορά ενώ στο Γυμνάσιο είχε τον ίδιο ποσοστό με τον αλγόριθμο επιλογής.

Τα αποτελέσματα αυτά επαληθεύουν την υπόθεση ότι ο αλγόριθμος της ευθείας ανταλλαγής δεν προκύπτει φυσικά από τις εμπειρίες των μαθητών οπότε δεν μπορεί να χτιστεί πάνω σε προϋπάρχουσα γνώση. Για αυτό κατά τη διδασκαλία του παρουσιάζονται πολλά προβλήματα (Astrachan, 2003· Geller & Dios, 1998· Simon et al., 2006· Βραχνός & Τζιμογιάννης 2014α). Επίσης επιβεβαιώνεται η τάση των

μαθητών να καταχωρούν τα ταξινομημένα στοιχεία σε μια νέα βοηθητική δομή ξεχωρίζοντάς τα από τα μη ταξινομημένα, κάτι που ήταν πολύ εύκολο στο υπολογιστικό μοντέλο της νοητής μηχανής που ορίσαμε.

Τα αποτελέσματα της έρευνας θεωρούμε ότι είναι χρήσιμα για τη διάρθρωση του προγράμματος σπουδών εισαγωγικών μαθημάτων αλγορίθμων και προγραμματισμού στην ενότητα της ταξινόμησης, αφού μπορούν να χρησιμοποιηθούν για την επιλογή και την διδακτική προσέγγιση των κατάλληλων αλγορίθμων ταξινόμησης που μπορούν να διδαχθούν οι μαθητές. Επίσης κάποια από τα παραπάνω συμπεράσματα μπορούν να αξιοποιηθούν στον σχεδιασμό οπτικοποιήσεων αλγορίθμων ταξινόμησης.

# 5

# *Μελέτη των δυσκολιών και των παρανοήσεων μαθητών για την έννοια του πίνακα*

## *5.1 Σκοπός και ερευνητικά ερωτήματα*

Ο πίνακας αποτελεί μια από τις θεμελιώδεις δομές δεδομένων της πληροφορικής και σε πολλές περιπτώσεις χρησιμοποιείται για την υλοποίηση πιο σύνθετων δομών δεδομένων όπως είναι οι στοίβα, η ουρά, το δέντρο και ο γράφος. Σύμφωνα με μια μεγάλη έρευνα που έγινε σε εκπαιδευτικούς πληροφορικής (Dale, 2006), ο πίνακας αποτελεί μια από τις δυσκολότερες έννοιες σε ένα εισαγωγικό μάθημα πληροφορικής για τους μαθητές. Αναφορές για παρανοήσεις και δυσκολίες που αντιμετωπίζουν οι μαθητές στους πίνακες υπάρχουν σε διάφορες έρευνες. Μια κλασική παρανόηση των μαθητών είναι ότι δυσκολεύονται να διακρίνουν τη διαφορά μεταξύ της έννοιας του δείκτη/θέσης/αναφοράς ενός στοιχείου του πίνακα και της τιμής του στοιχείου αυτού π.χ. 3, A[3] (du Boulay,1986· Lister et al., 2006).

Παρόλα αυτά δεν υπάρχει κάποια έρευνα η οποία να μελετά αποκλειστικά τις δυσκολίες των μαθητών στον προγραμματισμό με πίνακες.

Τα ερευνητικά ερωτήματα που οδήγησαν στον σχεδιασμό και την υλοποίηση αυτής της έρευνας είναι :

1. Ποιες είναι οι αναπαραστάσεις των μαθητών για την έννοια του πίνακα;
2. Ποιες είναι οι βασικές παρανοήσεις των μαθητών σχετικά με την έννοια του πίνακα;

3. Σε ποιο βαθμό οι μαθητές έχουν αναπτύξει την ικανότητα να επιλύουν αλγοριθμικά προβλήματα με τη χρήση πινάκων; Μπορούν να διακρίνουν σε ποια προβλήματα είναι απαραίτητη η χρήση πίνακα και σε ποια όχι;
4. Μπορεί η ταξινομία SOLO να αποτελέσει ένα αποτελεσματικό πλαίσιο για την ανάλυση των αναπαραστάσεων των μαθητών σχετικά με την έννοια του πίνακα;

## *5.2 Μεθοδολογία της Έρευνας*

### *5.2.1 Πειραματικός σχεδιασμός*

Η έρευνα έγινε σε δύο γενικά λύκεια της ευρύτερης αστικής περιοχής της Αθήνας (Πετρούπολης και Ιλίου). Έλαβε χώρα ένα μήνα μετά τη διδασκαλία της σχετικής ενότητας του μαθήματος, που αναφέρεται στην επεξεργασία πινάκων. Οι μαθητές μέχρι τότε είχαν εξοικειωθεί με την έννοια της μεταβλητής, των δομών επιλογής και επανάληψης και με τη δομή δεδομένων του πίνακα στο πλαίσιο του μαθήματος "Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον" της τεχνολογικής κατεύθυνσης της Γ' τάξης του Γενικού Λυκείου, όπου του δίνεται η δυνατότητα να διατυπώσουν τις λύσεις τους σε ψευδογλώσσα ή στην γλώσσα προγραμματισμού ΓΛΩΣΣΑ,

Δεν υπήρξε καμία διδακτική παρέμβαση πριν τη διεξαγωγή της έρευνας. Οι μαθητές είχαν διδαχθεί τη σχετική ύλη από τους καθηγητές τους. Για τη συμπλήρωση του ερωτηματολογίου μας διατέθηκε μια διδακτική ώρα, κατά την οποία ήταν παρόντες ο ερευνητής μαζί με τον καθηγητή της κάθε τάξης. Το ερωτηματολόγιο ήταν επώνυμο, ώστε να έχουμε στη συνέχεια τη δυνατότητα να διερευνήσουμε ακόμη περισσότερο, μέσω συνεντεύξεων, τις απόψεις των μαθητών που θα είχαν ερευνητικό ενδιαφέρον.

### *5.2.2 Δείγμα*

Το δείγμα περιελάμβανε 102 μαθητές της Γ' τάξης της τεχνολογικής κατεύθυνσης. Η πλειονότητα των μαθητών (93%), δήλωσαν ότι έχουν πρόσβαση σε υπολογιστή στο σπίτι. Το 24% των μαθητών δήλωσε ότι δεν έχει αναπτύξει ποτέ πρόγραμμα σε κάποιο προγραμματιστικό περιβάλλον στον υπολογιστή στα πλαίσια του μαθήματος. Το 82% είχε κάνει μαθήματα προγραμματισμού, στο μάθημα της Πληροφορικής της Α' Λυκείου.

Στην ανάλυση που ακολουθεί, συμπεριλήφθηκαν οι απαντήσεις 90 μαθητών. Δώδεκα (12) ερωτηματολόγια αγνοήθηκαν, γιατί σε αυτά δεν απαντήθηκαν τα περισσότερα ερωτήματα–έργα της έρευνας.

### *5.2.3 Εργαλείο της έρευνας*

Το εργαλείο της έρευνας ήταν επώνυμο ερωτηματολόγιο με τη μορφή διαγνωστικού τεστ που κλήθηκαν να απαντήσουν γραπτά οι μαθητές. Οι τρεις πρώτες ερωτήσεις ήταν πολύ απλές ασκήσεις εκτέλεσης αλγορίθμων και δόθηκαν με σκοπό να εξοικειωθούν οι μαθητές με τη διαδικασία και να ενισχυθεί η αυτοπεποίθηση και η διάθεση συμμετοχής τους. Τα αποτελέσματα των ερωτήσεων αυτών δεν συμπεριελήφθησαν στην ανάλυση, αλλά θα πρέπει να σημειωθεί ότι απαντήθηκαν σωστά από όλους σχεδόν τους μαθητές.

Τα βασικά προβλήματα-έργα της έρευνας στα οποία κλήθηκαν να απαντήσουν οι μαθητές ήταν τρία προγραμματιστικά προβλήματα ανοικτού τύπου. Ζητήθηκε από τους μαθητές να δώσουν σύντομη αλλά επαρκή εξήγηση-αιτιολόγηση για κάθε λύση που προτείνουν. Τα προβλήματα διατυπώθηκαν με τρόπο ώστε να ανταποκρίνονται στις βασικές προγραμματιστικές δεξιότητες που αναμένεται να αναπτύξουν οι μαθητές. Για το λόγο αυτό, υιοθετήθηκε η ιεραρχία τριών επιπέδων που προτάθηκε από τους Lister et al. (2009). Σύμφωνα με αυτή, στο χαμηλότερο επίπεδο βρίσκεται η ανάγνωση και εκτέλεση κώδικα εντολή προς εντολή. Στο ενδιάμεσο επίπεδο βρίσκεται η ικανότητα των μαθητών να εξηγήσουν τη λειτουργία που επιτελεί ένα τμήμα κώδικα. Τέλος, στο υψηλότερο επίπεδο βρίσκεται η ικανότητα της συγγραφής ενός προγράμματος για την επίλυση ενός πρωτότυπου προβλήματος ή της βελτίωσης-επέκτασης ενός προγράμματος. Τα τρία προβλήματα που δόθηκαν κατατάσσονται στις δύο υψηλότερες κατηγορίες.

Η κατηγορία που μας ενδιαφέρει περισσότερο είναι η μεσαία, όπου εξετάζουμε αν οι μαθητές μπορούν να περιγράψουν σε ένα αφηρημένο επίπεδο τη λειτουργία που επιτελεί ένα τμήμα αλγορίθμου. Δηλαδή αν μπορούν "να διακρίνουν το δάσος και όχι μόνο τα δέντρα" , όπως πολύ εύστοχα αναφέρεται στην εργασία των Lister et al.(2009).

## *5.3 Ανάλυση των Αποτελεσμάτων*

### *5.3.1 Έργο 1*

*Δίνονται δύο τμήματα αλγορίθμου, τα οποία διαβάζουν τους βαθμούς 100 μαθητών, με αποδεκτές τιμές στο διάστημα [1,20], και τους καταχωρούν σε έναν πίνακα. Ποιο από τα παραπάνω τμήματα αλγορίθμου θα επέλεγες να γράψεις εσύ; Να αιτιολογήσεις την απόφασή σου.*

| *Τμήμα Αλγορίθμου 1* | *Τμήμα Αλγορίθμου 2* |
|---|---|
| **Για** i **από** 1 **μέχρι** N<br>**Αρχή_Επανάληψης**<br>**Διάβασε** βαθ<br>**Μέχρις_ότου** βαθ>=1 **και** βαθ<=20<br>B[i] ← βαθ<br>**Τέλος_Επανάληψης** | **Για** i **από** 1 **μέχρι** N<br>**Αρχή_Επανάληψης**<br>**Διάβασε** B[i]<br>**Μέχρις_ότου** B[i]>=1 **και** B[i]<=20<br>**Τέλος_Επανάληψης** |

Το ερώτημα αυτό στοχεύει να διερευνήσει κατά πόσο η χρήση δομής πίνακα για τη διαχείριση και επεξεργασία δεδομένων του ίδιου τύπου γίνεται από τους μαθητές αιτιολογημένα. Στον Πίνακα 5.1 δίνονται οι απαντήσεις των μαθητών και η κατά SOLO ταξινόμησή τους. Το 36.7% των μαθητών έδωσαν απαντήσεις οι οποίες κατατάχθηκαν στο προδομικό επίπεδο. Κάποιες από αυτές δεν είχαν σχέση με το ερώτημα ή έδειχναν σοβαρές ελλείψεις και δυσκολίες σε βασικές προγραμματιστικές γνώσεις και δεξιότητες όπως οι παρακάτω ενδεικτικές απαντήσεις:

*«Δεν καταλαβαίνω το πρόβλημα.»*

*«Η συνθήκη εγκυρότητας είναι λάθος.»*

Μια δεύτερη κατηγορία απαντήσεων που δόθηκαν από 18 μαθητές, κατατάχθηκαν επίσης στο προδομικό επίπεδο. Οι απαντήσεις αυτές σχετίζονταν με ανάκληση ή απομνημόνευση τμήματος κώδικα που οι μαθητές χρησιμοποιούν συχνά σε ασκήσεις χωρίς όμως να κατανοούν τη λειτουργία του. Μερικές χαρακτηριστικές απαντήσεις μαθητών είναι οι παρακάτω:

*«Ο αλγόριθμος Α1 μου φαίνεται πιο σωστός γιατί αυτό χρησιμοποιώ και εγώ.»*

*«Ο αλγόριθμος Α1 είναι ο σωστός τρόπος ελέγχου εγκυρότητας.»*

**Πίνακας 5.1. Απαντήσεις των μαθητών στο πρώτο πρόβλημα**

| Επίπεδο SOLO | Ποσοστό % | Αριθμός Μαθητών (N=90) |
|---|---|---|
| Προδομικό | 36.7 | 33 |
| Μονοδομικό | 0 | 0 |
| Πολυδομικό | 28.9 | 26 |
| Συνθετικό | 34.4 | 31 |

Στο πολυδομικό επίπεδο έχουμε κατατάξει τις απαντήσεις 26 μαθητών οι οποίοι περιγράφουν σωστά όλο το τμήμα κώδικα γραμμή προς γραμμή, χωρίς να επισημάνουν τον τρόπο με τον οποίο συνδέονται αυτές οι εντολές. Δηλαδή δεν ασχολούνται με την σύνδεση του ελέγχου εγκυρότητας με την απόδοση τιμής στα στοιχεία του πίνακα. Χαρακτηριστικές απαντήσεις είναι οι παρακάτω:

*«Επέλεξα τον αλγόριθμο Α1 επειδή ο έλεγχος εγκυρότητας πρέπει να γίνει πριν την εντολή εκχώρησης της τιμής στο στοιχείο του πίνακα.»*

Οι περισσότεροι μαθητές απέφυγαν να χρησιμοποιήσουν το στοιχείο του πίνακα απευθείας στον έλεγχο της συνθήκης εγκυρότητας. Προτίμησαν να χρησιμοποιήσουν μια επιπλέον (βοηθητική) μεταβλητή για την εισαγωγή των δεδομένων και στη συνέχεια μόνο όταν τα δεδομένα είναι έγκυρα, να τα καταχωρήσουν στον πίνακα.

Από την ανάλυση των απαντήσεων προέκυψε ότι οι μαθητές παρουσιάζουν δυσκολίες στην αποτελεσματική χρήση πινάκων. Μια παρανόηση που φαίνεται να έχουν οι μαθητές είναι ότι δεν χρησιμοποιούν ένα στοιχείο πίνακα (G[i]) με τον ίδιο τρόπο με τον οποίο χρησιμοποιούν μια μεταβλητή. Η αναπαράσταση που έχουν για το στοιχείο του πίνακα δεν σχετίζεται με την έννοια της μεταβλητής. Για το λόγο αυτό οι μαθητές αποφεύγουν να χρησιμοποιήσουν το στοιχείο του πίνακα (G[i]) στη συνθήκη εγκυρότητας.

Τέλος, το 34% των απαντήσεων των μαθητών κατατάχθηκε στο συνθετικό επίπεδο. Χαρακτηριστικές απαντήσεις αυτού του επιπέδου αφαίρεσης είναι οι παρακάτω:

*«Ο αλγόριθμος Α2 απαιτεί λιγότερη μνήμη.»*

*«Ο αλγόριθμος Α2 είναι πιο αποδοτικός αφού δεν χρησιμοποιεί επιπλέον μεταβλητή.»*

Στις παραπάνω απαντήσεις φαίνεται ότι οι μαθητές έχουν οικοδομήσει αποτελεσματικά νοητικά μοντέλα για τις έννοιες του πίνακα και της μεταβλητής, τα

οποία τους επιτρέπουν να διακρίνουν χαρακτηριστικά του αλγορίθμου πέρα από τη βασική λειτουργία του, όπως είναι η απόδοση αλγορίθμου και η έννοια της προσωρινής μνήμης.

### *5.3.2 Έργο 2*

*Ο παρακάτω αλγόριθμος υπολογίζει τον μεγαλύτερο από τους βαθμούς 50 μαθητών μιας τάξης. Να τροποποιήσεις τον αλγόριθμο έτσι ώστε να υπολογίζει και να εμφανίζει πόσοι μαθητές έχουν τον μεγαλύτερο βαθμό (Στην γενική περίπτωση, μπορούν να υπάρχουν περισσότεροι του ενός μαθητές που έχουν τον μεγαλύτερο βαθμό).*

```
Διάβασε βαθ
max ← βαθ
Για μαθητή από 2 μέχρι 50
    Διάβασε βαθ
    Αν βαθ > max Τότε
        max ← βαθ
    Τέλος_Αν
Τέλος_Επανάληψης
```

Το έργο αυτό έχει στόχο να διερευνήσει κατά πόσο οι μαθητές μπορούν να αντιληφθούν την αναγκαιότητα χρήσης πίνακα σε ένα πρόβλημα επεξεργασίας δεδομένων. Ο αλγόριθμος πρώτα υπολογίζει τη μέγιστη τιμή και στη συνέχεια ελέγχει πόσοι βαθμοί είναι ίσοι με αυτή. Πρόκειται για ένα πρόβλημα που είναι στο πνεύμα του μαθήματος αποτελεί παράδειγμα διδασκαλίας και ζητείται συχνά σε εξετάσεις. Η αναγκαιότητα της χρήσης πίνακα προκύπτει εδώ από το γεγονός ότι θα πρέπει πρώτα να υπολογιστεί η μέγιστη τιμή και μετά να ελεγχθούν όλα τα στοιχεία. Θα πρέπει να σημειωθεί ότι μπορεί να σχεδιαστεί αλγόριθμος, που να υπολογίζει το πλήθος των μαθητών με τον μεγαλύτερο βαθμό χωρίς τη χρήση πίνακα, αλλά είναι αρκετά δύσκολος στη σύλληψη. Η λύση αυτή, που παρουσιάζεται στο Τμήμα Κώδικα 5.1, δεν δόθηκε από κανέναν μαθητή.

```
Διάβασε βαθμός
max ← βαθμός
φορές ← 1
Για i από 2 μέχρι 50
    Διάβασε βαθμός
    Αν βαθμός > max Τότε
        max ← βαθμός
        φορές ← 1
    Αλλιώς_Αν βαθμός = max Τότε
        φορές ← φορές + 1
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 5.1. Υπολογισμός συχνότητας μέγιστης τιμής χωρίς χρήση πίνακα**

Στον Πίνακα 5.2 παρουσιάζονται τα αποτελέσματα των απαντήσεων των μαθητών σε σχέση με τα επίπεδα της ταξινομίας SOLO.

**Πίνακας 5.2. Οι απαντήσεις των μαθητών στο 2ο πρόβλημα**

| Επίπεδο SOLO | Ποσοστό % | Αριθμός μαθητών (N=90) |
|---|---|---|
| Δεν απάντησαν | 11.1 | 10 |
| Προδομικό | 12.2 | 11 |
| Μονοδομικό | 22.2 | 20 |
| Πολυδομικό | 30 | 27 |
| Συνθετικό | 24.4 | 22 |

Στο προδομικό επίπεδο έχουμε κατατάξει απαντήσεις στις οποίες οι μαθητές απλά έχουν αντιγράψει τον αλγόριθμο που τους δόθηκε και έχουν προσθέσει έναν μετρητή τον οποίο όμως δεν χρησιμοποιούν σωστά αφού ενώ του δίνουν αρχική τιμή δεν το αυξάνουν σε κάθε επανάληψη.

```
Διάβασε βαθμός
max ← βαθμός
φορές ← 0
Για i από 2 μέχρι 50
    Διάβασε βαθμός
    Αν βαθμός > max Τότε
        max ← βαθμός
        Γράψε φορές + 1
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 5.2. Εσφαλμένη χρήση μεταβλητής-μετρητή**

Στο μονοδομικό επίπεδο κατατάχθηκαν παρόμοιες απαντήσεις με αυτές του προδομικού με τη διαφορά ότι εδώ οι μαθητές χρησιμοποιούν τον μετρητή σωστά. Δείχνουν δηλαδή ότι έχουν κατανοήσει τη λειτουργία που επιτελεί ένα μικρό τμήμα κώδικα. Πιο συγκεκριμένα υπολογίζουν σωστά τη μέγιστη τιμή, αυξάνουν τον μετρητή αλλά σε λάθος σημείο, που σημαίνει ότι δεν έχουν οικοδομήσει ολοκληρωμένες αναπαραστάσεις για ολόκληρο το τμήμα κώδικα.

```
Διάβασε βαθμός
max ← βαθμός
φορές ← 0
Για i από 2 μέχρι 50
    Διάβασε βαθμός
    Αν βαθμός > max Τότε
        max ← βαθμός
        φορές ← φορές + 1
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 5.3. Απάντηση στο μονοδομικό επίπεδο της ταξινομίας SOLO**

Στο πολυδομικό επίπεδο αντιστοιχίστηκαν οι απαντήσεις από τις οποίες φαίνεται ότι οι μαθητές έχουν κατανοήσει τη λογική του κώδικα και ο αλγόριθμος που έχουν σχεδιάσει δίνει μια ολοκληρωμένη λύση που έχει κάποια λογική. Ωστόσο οι μαθητές σε αυτή την περίπτωση δεν διέκριναν την ανάγκη για αποθήκευση όλων των δεδομένων στη μνήμη χρησιμοποιώντας μια δομή όπως αυτή του πίνακα.

```
Διάβασε βαθμός
max ← βαθμός
Για i από 2 μέχρι 50
    Διάβασε βαθμός
    Αν βαθμός > max Τότε
        max ← βαθμός
    Τέλος_Αν
Τέλος_Επανάληψης
φορές ← 0
Για i από 1 μέχρι 50
    Αν βαθμός = max Τότε
        φορές ← φορές + 1
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 5.4. Παρανόηση ιστορικού τιμών μεταβλητής (stack misconception)**

Είναι φανερό ότι οι μαθητές που έδωσαν την παραπάνω λύση πιστεύουν ότι η μεταβλητή βαθμός διατηρεί όλες τις τιμές που έχει πάρει μέχρι εκείνη τη στιγμή. Αυτό πιστεύουμε ότι οφείλεται σε λανθασμένη αναπαράσταση που έχουν για την έννοια της μεταβλητής. Οι μαθητές υιοθετούν ένα μοντέλο στοίβας ή σωρού (Jimoyiannis, 2011) για την αναπαράσταση μιας απλής μεταβλητής στο οποίο όχι μόνο διατηρούνται όλες οι τιμές που έχει πάρει η μεταβλητή αλλά μπορούμε να τις ανακτήσουμε με κάποιον ‘μαγικό’ τρόπο, αφού με απλή αναφορά του ονόματος της μεταβλητής η νοητή μηχανή καταλαβαίνει σε ποια τιμή του ‘ιστορικού τιμών’ θέλουμε να γίνει προσπέλαση. Εξαιτίας αυτής της παρανόησης, οι μαθητές δεν κατάφεραν να διακρίνουν την ανάγκη χρήσης πίνακα για την επίλυση του προβλήματος αυτού. Το παράδειγμα αυτό είναι χαρακτηριστικό και αναδεικνύει την επίδραση που έχουν οι παρανοήσεις των μαθητών για την έννοια της μεταβλητής σε πιο σύνθετες έννοιες που οικοδομούνται πάνω στη μεταβλητή, όπως είναι η έννοια του πίνακα.

Στο πολυδομικό επίπεδο κατατάξαμε 8 ακόμη απαντήσεις μαθητών στις οποίες αντί να χρησιμοποιηθεί πίνακας, γίνεται επανεισαγωγή των ίδιων δεδομένων. Και σε αυτή την περίπτωση οι μαθητές δεν μπορούν να διακρίνουν την αναγκαιότητα χρήσης πίνακα. Όμως, φαίνεται ότι οι μαθητές αυτοί έχουν οικοδομήσει επαρκείς αναπαραστάσεις για την έννοια της μεταβλητής. Δηλαδή έχουν κατανοήσει ότι σε μια μεταβλητή αποθηκεύεται κάθε φορά μια ακριβώς τιμή, κάτι που σημαίνει ότι όλα τα

δεδομένα εκτός της τελευταίας τιμής, έχουν χαθεί. Όμως για να αποφύγουν αυτή τη δυσκολία οι μαθητές θεώρησαν ότι γίνεται επανεισαγωγή των δεδομένων κάτι που δεν αναφέρεται στην εκφώνηση. Ουσιαστικά οι μαθητές με την αυθαίρετη αυτή υπόθεση έλυσαν διαφορετικό πρόβλημα, επειδή τους διευκόλυνε.

Στις απαντήσεις που χαρακτηρίστηκαν ως συνθετικές (24%) οι μαθητές παρουσίασαν μια ολοκληρωμένη και σωστή λύση με κατάλληλη χρήση πίνακα και μετρητή, όπως φαίνεται στο Τμήμα Κώδικα 5.5.

```
Διάβασε βαθμός[1]
max ← βαθμός[1]
Για i από 2 μέχρι 50
    Διάβασε βαθμός[i]
    Αν βαθμός[i] > max Τότε
        max ← βαθμός[i]
    Τέλος_Αν
Τέλος_Επανάληψης
φορές ← 0
Για i από 1 μέχρι 50
    Αν βαθμός[i] = max Τότε
        φορές ← φορές + 1
    Τέλος_Αν
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 5.5. Υπολογισμός συχνότητας μέγιστης τιμής με χρήση πίνακα.**

Οι παραπάνω απαντήσεις θα μπορούσαν να καταταχθούν στο υψηλότερο επίπεδο της εκτεταμένης αφαίρεσης, διότι οι μαθητές χρησιμοποιούν μια νέα δομή για να λύσουν το πρόβλημα. Ωστόσο τις κατατάξαμε στο συνθετικό επίπεδο διότι οι μαθητές δεν σχεδίασαν από την αρχή όλη τη λύση αλλά κλήθηκαν να τροποποιήσουν μια υπάρχουσα λύση.

### *5.3.3 Έργο 3*

*Δίνεται ο πίνακας ακέραιων αριθμών Χ με τιμές όπως παρακάτω:*

| | | | | | |
|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | 1 |

*Τι περιμένετε να εμφανιστεί στην οθόνη κατά την εκτέλεση του παρακάτω προγράμματος; Ποιες θα είναι οι τελικές τιμές των στοιχείων του πίνακα; Να αιτιολογήσετε την απάντησή σας.*

```
i ← 1
Όσο i < X[i] Επανάλαβε
    X[X[i]] ← X[i]
    i ← i + 1
    Εμφάνισε i, X[i], X[X[i]]
Τέλος_Επανάληψης
```

Η δομή του παραπάνω αλγορίθμου είναι αρκετά απλή, αφού αποτελείται από μια δομή επανάληψης η οποία περιέχει δύο εντολές εκχώρησης. Παρόλα αυτά το πρόβλημα αυτό αποδείχθηκε το πιο δύσκολο για τους μαθητές αφού απάντησε σωστά μόνο το 11% του δείγματος (Πίνακας 5.3). Πολλοί μαθητές (30%) δήλωσαν ότι αδυνατούν να εκτελέσουν τον αλγόριθμο βήμα-βήμα και κυρίως ότι δυσκολεύονται να κατανοήσουν την έκφραση X[X[i]], για αυτό και δεν έδωσαν καμία απάντηση.

**Πίνακας 5.3. Απαντήσεις των μαθητών στο έργο 3**

| Επίπεδο SOLO | Ποσοστό % | Αριθμός μαθητών (N=90) |
|---|---|---|
| Δεν απάντησαν | 33.3 | 30 |
| Προδομικό | 16.7 | 15 |
| Μονοδομικό | 38.9 | 35 |
| Πολυδομικό | 11.1 | 10 |

Σύμφωνα με τους Jakwerth & Stancavage (2003) υπάρχουν τρεις βασικοί λόγοι που οι μαθητές αφήνουν αναπάντητες ερωτήσεις: α) δυσκολίες στην κατανόηση του προβλήματος β) ελλιπείς γνώσεις – δυσκολίες σχετικά με το πρόβλημα και γ) έλλειψη χρόνου. Στην δική μας έρευνα οι μαθητές είχαν αρκετό χρόνο για να απαντήσουν στα προβλήματα που δόθηκαν. Η ερμηνεία που δίνουμε στο γεγονός ότι ένας στους τρεις μαθητές δεν προσπάθησε να δώσει απάντηση ήταν η δυσκολία του συγκεκριμένου προβλήματος, η οποία σχετίζεται με τον χειρισμό της έκφρασης X[ X[ i ] ].

Στο προδομικό επίπεδο κατατάξαμε απαντήσεις στις οποίες φαίνεται ότι οι μαθητές διατηρούν ημιτελείς ή λανθασμένες αναπαραστάσεις για τις έννοιες του

πίνακα και του δείκτη πίνακα (16.7%), όπως φαίνεται παρακάτω:

*«Δεν μπορεί να μπει το Χ[i] μέσα σε αγκύλες. Εκεί μπαίνει δείκτης.»*

«*Το Χ[i] δεν μπορεί να είναι δείκτης πίνακα.*»

«*Το Χ[i] είναι πίνακας και όχι μεταβλητή.*»

Αρκετοί μαθητές πιστεύουν ότι το στοιχείο Χ[i] είναι ένας πίνακας και όχι μια μεταβλητή. Αυτή είναι μια ένδειξη που μας οδηγεί στο συμπέρασμα ότι η δυσκολία των μαθητών να χρησιμοποιήσουν το στοιχείο πίνακα Χ[i] στη θέση του δείκτη του πίνακα οφείλεται στο γεγονός ότι δεν έχουν κατανοήσει ότι το στοιχείο Χ[i] είναι ουσιαστικά μια μεταβλητή και μπορεί να χρησιμοποιηθεί όπως μια απλή μεταβλητή.

Επίσης στο προδομικό επίπεδο έχουμε κατατάξει τις απαντήσεις που δόθηκαν από επτά (7) μαθητές, στις οποίες αναφέρονται στο στοιχείο Χ[i] ως πίνακα. Σε αυτή την περίπτωση οι μαθητές έχουν σχηματίσει την λανθασμένη αναπαράσταση ότι το στοιχείο Χ[i] αναφέρεται σε όλο τον πίνακα Χ.

Ενδέχεται τα σχολικά παραδείγματα και ο φορμαλισμός που χρησιμοποιείται στα εγχειρίδια ή/και στη διδασκαλία στην τάξη να συμβάλλουν στην παρανόηση αυτή.

Στα εισαγωγικά μαθήματα προγραμματισμού είναι κοινή πρακτική από τους εκπαιδευτικούς να χρησιμοποιούν απλά σύμβολα για την ονοματολογία πινάκων (A, B, X) και για τους δείκτες των στοιχείων (i, j). Τα σύμβολα αυτά δίνουν ένα μαθηματικό φορμαλισμό στα προβλήματα και συνιστούν μια πηγή δυσκολιών για τους μαθητές.

Στο μονοδομικό επίπεδο κατατάξαμε το 38.9% των απαντήσεων στις οποίες οι μαθητές παρουσίασαν μεγάλη δυσκολία στην σύνδεση της έννοιας του δείκτη ενός στοιχείου του πίνακα με την τιμή του στοιχείου αυτού. Οι μαθητές αντιμετώπισαν το στοιχείο Χ[i] ως μια στατική μεταβλητή η οποία έχει μια συγκεκριμένη θέση στη μνήμη, κάτι το οποίο όμως δεν ισχύει γιατί με την αλλαγή του δείκτη i αλλάζει και το στοιχείο Χ[i] στο οποίο αναφερόμαστε. Στον πίνακα 5.4 δίνεται ένα χαρακτηριστικό παράδειγμα πίνακα τιμών που δόθηκε από μαθητή, και στο οποίο φαίνεται ότι τα στοιχεία Χ[i], Χ[Χ[i]] αντιμετωπίζονται ως στατικές μεταβλητές.

**Πίνακας 5.4. Λανθασμένος πίνακας τιμών εκτέλεσης του αλγορίθμου για το έργο 3**

| i | X[i] | X[X[i]] | Έξοδος |
|---|---|---|---|
| 1 | 6 | 6 | 2 6 6 |
| 2 | 5 | 5 | 3 5 5 |
| 3 | 4 | 3 | 4 4 3 |
| 4 | 3 | 5 | |

Από τον Πίνακα 5.4, είναι πλέον φανερό ότι όταν οι μαθητές ενημερώνουν την τιμή της μεταβλητής i, π.χ. από i=1 σε i=2, δεν ενημερώνουν αντίστοιχα τη μεταβλητή X[i] με αποτέλεσμα αυτή να συνεχίσει να αναφέρεται στην X[1]. Έτσι η εντολή εξόδου που ακολουθεί εμφανίζει στην οθόνη τις τιμές 2, X[1] and X[X[1]] αντί για 2, X[2] and X[X[2]]. Οι απαντήσεις αυτής της κατηγορίας αποκαλύπτουν ένα λανθασμένο στατικό μοντέλο της προγραμματιστικής έκφρασης X[i] όσον αφορά την δυναμική φύση της σχέσης του δείκτη i με την αντίστοιχη τιμή του στοιχείου X[i].

Επίσης, από τον Πίνακα 5.4 φαίνεται ότι δεν έχουν κατανοήσει ότι κατά την εκτέλεση του αλγορίθμου το στοιχείο X[i] δεν αναφέρεται σε ένα συγκεκριμένο στοιχείο του πίνακα, αλλά σε πολλά, ανάλογα με την τιμή του i, π.χ. πρώτα αναφέρεται στο X[1], μετά στο X[2], κ.ο.κ.

Η σχέση μεταξύ του δείκτη i και του αντίστοιχου στοιχείου X[i] μοιάζει πολύ με τη σχέση της μεταβλητής x με την τιμή της πραγματικής συνάρτησης f(x). Φυσικά στην περίπτωση του πίνακα η συνάρτηση ορίζεται στους φυσικούς αριθμούς, και παραπέμπει περισσότερο στην έννοια της ακολουθίας $a_n$.

Σύμφωνα με τους Carlson (1998) και Thomson (1994), οι μαθητές αδυνατούν να οικοδομήσουν τις εννοιολογικές δομές οι οποίες θα τους επιτρέψουν τη μοντελοποίηση των συναρτησιακών σχέσεων στις οποίες η τιμή της συνάρτησης f(x) αλλάζει όποτε αλλάζει η μεταβλητή x. Επίσης οι μαθητές παρουσιάζουν σημαντικές δυσκολίες σε πιο σύνθετες συναρτησιακές έννοιες όπως είναι η σύνθεση συναρτήσεων, π.χ. f(f(x)). Στη συγκεκριμένη περίπτωση υπάρχει μια συναρτησιακή αναπαράσταση ή αναλογία στη σκέψη των μαθητών σχετικά με την έκφραση X[X[i]], για παράδειγμα το στοιχείο X[i] εξαρτάται από την τιμή του i. Έτσι, οι μαθητές δεν μπορούν να ενημερώσουν τις εκφράσεις X[i] και X[X[i]] με την αλλαγή του i, αφού δεν έχουν κατανοήσει αυτή τη δυναμική σχέση.

Τέλος, μόνο το 11% των μαθητών έδωσε σωστή απάντηση η οποία κατηγοριοποιήθηκε στο πολυδομικό επίπεδο. Οι μαθητές αυτοί φαίνεται να έχουν κατανοήσει τη δυναμική φύση της σχέσης μέσα από τον τρόπο με τον οποίο εκτέλεσαν τον αλγόριθμο εντολή-εντολή. Επίσης διέκριναν ότι η μεταβλητή X[i] στην εντολή εξόδου Εμφάνισε έχει αλλάξει. Αυτή η απάντηση δείχνει ότι οι μαθητές έχουν κατανοήσει πλήρως τη δυναμική σχέση μεταξύ της μεταβλητής i και του στοιχείου X[i] του πίνακα. Στον πίνακα 5.5 δίνεται ένα χαρακτηριστικό παράδειγμα πολυδομικής απάντησης.

**Πίνακας 5.5. Πίνακας τιμών εκτέλεσης του αλγορίθμου για το έργο 3**

| i | X[1] | X[2] | X[3] | X[4] | X[5] | X[6] | X[X[i]] ← X[i] | Εμφάνισε i, X[i], X[X[i]] |
|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 5 | 4 | 3 | 2 | 1 | | |
| | 6 | 5 | 4 | 3 | 2 | **6** | X[6] ← X[1] | |
| 2 | 6 | 5 | 4 | 3 | 2 | 6 | | 2 5 2 |
| | 6 | 5 | 4 | 3 | **5** | 6 | X[5] ← X[2] | |
| 3 | 6 | 5 | 4 | 3 | 5 | 6 | | 3 4 3 |
| | 6 | 5 | 4 | **4** | 5 | 6 | X[4] ← X[3] | |
| 4 | 6 | 5 | 4 | 4 | 5 | 6 | | 4 4 4 |

## *5.4 Συζήτηση*

Η ανάλυση των αποτελεσμάτων έδειξε ότι οι μαθητές παρουσιάζουν σημαντικές δυσκολίες στην κατανόηση και χρήση της δομής του πίνακα. Οι περισσότεροι μαθητές έδωσαν απαντήσεις οι οποίες κατατάχθηκαν στα χαμηλότερα επίπεδα της ταξινομίας SOLO. Το κύριο συμπέρασμα της έρευνας ήταν ότι οι μαθητές έχουν οικοδομήσει ημιτελείς και σε κάποιες περιπτώσεις λανθασμένες αναπαραστάσεις για την έννοια του πίνακα και το ρόλο του δείκτη στοιχείου πίνακα. Αυτές δεν τους επιτρέπουν να διατυπώσουν επαρκή αλγόριθμο επίλυσης προβλημάτων στις περιπτώσεις που πρέπει να συσχετίσουν τις έννοιες του πίνακα και του δείκτη πίνακα. Επίσης φαίνεται ότι κάποιες από τις παρανοήσεις των μαθητών για τις μεταβλητές εδράζονται σε προηγούμενες παρανοήσεις στην έννοια της μεταβλητής.

Η πλειονότητα των μαθητών παρουσίασαν δυσκολίες στην επίλυση προβλημάτων με πίνακες. Η πρώτη δυσκολία εντοπίστηκε κατά την ανάλυση των απαντήσεων των

μαθητών στο πρώτο πρόβλημα που τους τέθηκε, όπου φάνηκε ότι οι μαθητές δεν είναι εξοικειωμένοι με τη χρήση ενός στοιχείου του πίνακα ως απλή μεταβλητή. Φαίνεται ότι πολλοί μαθητές αντιλαμβάνονται τον πίνακα ως μια ενιαία δομή ένα αντικείμενο και όχι ως ένα διατεταγμένο σύνολο μεταβλητών. Οι μαθητές αυτοί έχουν δυσκολίες να κατανοήσουν ότι ένα στοιχείο πίνακα και ο δείκτης του μπορούν να χρησιμοποιηθούν και να συμμετέχουν σε αλγοριθμικές εκφράσεις ή υπολογισμούς όπως ακριβώς και μια απλή αριθμητική μεταβλητή.

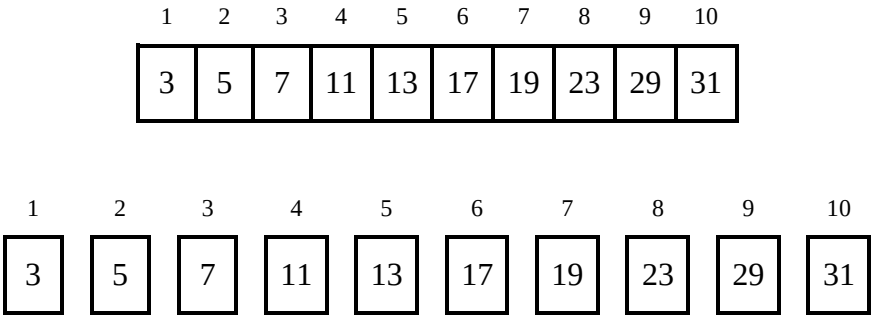
Η ίδια παρανόηση καταγράφηκε και στην ανάλυση των απαντήσεων του τρίτου προβλήματος, όπου το 90% των μαθητών έδειξαν ότι δεν έχουν κατανοήσει πως όταν αλλάζει η μεταβλητή i, αλλάζει και το στοιχείο X[i] στο οποίο αναφέρεται. Το πρόβλημα αυτό έδειξε ότι οι μαθητές αντιμετωπίζουν μεγάλες δυσκολίες όταν έρχονται αντιμέτωποι με εκφράσεις όπως X[X[i]], τις οποίες δεν μπορούν να χειριστούν αποτελεσματικά. Σε αυτό το πρόβλημα αρκετοί μαθητές έκαναν την παρατήρηση ότι ένα στοιχείο πίνακα δεν μπορεί να χρησιμοποιηθεί ως δείκτης. Αυτό μπορεί να ερμηνευθεί ως επέκταση της παρανόησης που έχουν οι μαθητές ότι ένα στοιχείο πίνακα δεν μπορεί να χρησιμοποιηθεί όπως μια μεταβλητή.

Για το θέμα αυτό υπάρχουν πολύ λίγες αναφορές στη βιβλιογραφία. Οι Hazzan, Lapidot & Ragonis (2011) αναφέρουν ότι κατά την εκπόνηση ενός σεναρίου διδασκαλίας για τους πίνακες, ένας από τους βασικούς διδακτικούς στόχους είναι να κατανοήσουν οι μαθητές ότι ένα στοιχείο πίνακα είναι ουσιαστικά μια μεταβλητή. Πιο πρόσφατα, οι Craig & Petersen (2016), στην έρευνά τους για τις δυσκολίες των φοιτητών στους δείκτες της C βρήκαν ότι οι φοιτητές προτιμούν να χρησιμοποιούν μεταβλητές εκεί που θα μπορούσαν να χρησιμοποιήσουν απευθείας στοιχεία ενός πίνακα. Μια ερμηνεία για αυτό είναι ότι οι φοιτητές αποφεύγουν να χρησιμοποιούν το στοιχείο πίνακα ως μεταβλητή γιατί έχουν οικοδομήσει διαφορετικό νοητικό μοντέλο για την έννοια της μεταβλητής και διαφορετικό για την έννοια του στοιχείου του πίνακα.

Αυτό πιστεύουμε ότι οφείλεται στην διδακτική προσέγγιση που ακολουθούν οι εκπαιδευτικοί όταν εισάγουν τους μαθητές στους πίνακες. Σχεδόν σε όλα τα βιβλία ο πίνακας απεικονίζεται ως μια ακολουθία κελιών τα οποία είναι ενωμένα μεταξύ τους και παραπέμπουν περισσότερο σε ένα ενιαίο αντικείμενο παρά σε ένα σύνολο θέσεων στη μνήμη. Η γραφική αναπαράσταση του πίνακα τονίζει περισσότερο της ιδιότητες μιας ενιαίας δομής εις βάρος των ιδιοτήτων των δομικών στοιχείων της, αν αυτά

αντιμετωπιστούν ως μεμονωμένες οντότητες.

Στο Σχήμα 5.1 φαίνονται δυο διαφορετικές αναπαραστάσεις ενός πίνακα 10 στοιχείων. Στην πρώτη τα κελιά είναι ενωμένα ενώ στη δεύτερη υπάρχουν κενά ανάμεσά τους, έτσι ώστε να φαίνεται ότι πρόκειται για ξεχωριστές θέσεις στη μνήμη οι οποίες θα μπορούσαν να θεωρηθούν ως απλές μεταβλητές.




**Σχήμα 5.1. Γραφική αναπαράσταση ενός πίνακα**

Η αναπαράσταση που έχουν οι μαθητές για τον πίνακα ως ενιαίο αντικείμενο θα μπορούσε να θεωρηθεί ως ένα παράδειγμα λανθασμένης αναλογίας (Soloway & Spohrer 1989), όπου οι μαθητές προσπαθούν να εξάγουν περισσότερες δομές ή σχέσεις από μια αναλογία. Μια ανάλογη παρανόηση, η οποία οφείλεται επίσης σε λανθασμένη ερμηνεία ή μεταφορά πληροφορίας από μια αναπαραστατική αναλογία, αποτελεί η αντίληψη ότι μια μεταβλητή μπορεί να διατηρεί ταυτόχρονα περισσότερες από μια τιμές. Η αναλογία που ευθύνεται για αυτό είναι η γνωστή γραφική επεξήγηση της έννοιας της μεταβλητής με ένα γραμματοκιβώτιο το οποίο περιέχει ένα γράμμα. Στην πραγματικότητα ένα γραμματοκιβώτιο μπορεί να περιέχει περισσότερα από ένα γράμματα και αυτό το γνωρίζουν οι μαθητές. Κατά συνέπεια, ο συλλογισμός ότι και η μεταβλητή μπορεί να διατηρεί περισσότερες της μιας τιμές είναι συνέπεια της ερμηνείας της αναλογίας που, αναπόφευκτα, δίνουν αρκετοί μαθητές.

Την παρανόηση αυτή τη συναντήσαμε στο δεύτερο πρόβλημα το οποίο εξέταζε την ικανότητα των μαθητών να διακρίνουν αν για την επίλυση ενός προβλήματος χρειάζεται πίνακας. Μόνο το 24% των μαθητών χρησιμοποίησε πίνακα για να λύσει το πρόβλημα ενώ το 21% θεώρησε ότι μια μεταβλητή διατηρεί όλες τις προηγούμενες τιμές που έχει πάρει (παρανόηση ιστορικού).

Είναι φανερό ότι οι παρανοήσεις που έχουν οι μαθητές για την έννοια της μεταβλητής δημιουργούν νέες παρανοήσεις και δυσκολίες σε αλγορίθμους που σχετίζονται με πίνακες. Αυτό μπορεί να ερμηνευτεί, αν λάβουμε υπόψη το γεγονός ότι

η έννοια του πίνακα οικοδομείται πάνω στην έννοια της μεταβλητής, οπότε οι δυσκολίες που παρουσιάζουν οι μαθητές σε απλές δομικές έννοιες μεταφέρονται στις πιο σύνθετες.

Μια άλλη παρανόηση που εντοπίστηκε σε αυτή την έρευνα είναι ότι αρκετοί μαθητές όταν αναφέρονται σε έναν πίνακα Χ, χρησιμοποιούν την έκφραση Χ[i] και όχι το όνομα του πίνακα (Χ). Αυτό ενδεχομένως να οφείλεται στον ψευδοκώδικα που χρησιμοποιείται στο σχετικό μάθημα της αλγοριθμικής αφού οι μαθητές δεν χρησιμοποιούν ποτέ το όνομα του πίνακα για να αναφερθούν σε αυτόν, παρά μόνον όταν μεταβιβάζεται ως παράμετρος σε ένα υποπρόγραμμα. Ο τμηματικός προγραμματισμός όμως είναι η τελευταία ενότητα του μαθήματος, με αποτέλεσμα οι μαθητές να έχουν ήδη οικοδομήσει αναπαραστάσεις για την έννοια του πίνακα όταν έχουν φτάσει εκεί. Ένας άλλος λόγος που οι μαθητές αναφέρονται σε όλο τον πίνακα ως Χ[i], είναι ότι οι διδακτικές προσεγγίσεις που ακολουθούνται συχνά στα μαθήματα της αλγοριθμικής είναι περισσότερο λειτουργικές/χρηστικές και όχι εννοιολογικές. Δεν στοχεύουν δηλαδή στην κατανόηση και λειτουργική εφαρμογή των βασικών αλγοριθμικών εννοιών, αλλά κυρίως στην επίλυση προβλημάτων τα οποία μάλιστα είναι συγκεκριμένα και τυποποιούνται εύκολα.

```
Πρόγραμμα  Εύρεση_Μεγίστου
Μεταβλητές
   Ακέραιες: i, X[i], μέγιστο ! λάθος δήλωση του πίνακα Χ, αντί για Χ[10]
Αρχή
   Για i από 1 μέχρι 10
      Διάβασε X[i]
   Τέλος_Επανάληψης
   μέγιστο ← –1
   Για i από 1 μέχρι 10
      Αν X[i] > μέγιστο Τότε
         μέγιστο ← X[i]
      Τέλος_Αν
   Τέλος_Επανάληψης
   Γράψε μέγιστο
Τέλος_Προγράμματος
```

**Τμήμα Κώδικα 5.6. Υπολογισμός μέγιστης τιμής των στοιχείων ενός πίνακα**

Επίσης πολλές φορές οι μαθητές δηλώνουν έναν πίνακα σε ένα πρόγραμμα λανθασμένα ως X[i] όπως φαίνεται στο Τμήμα Κώδικα 5.6. Για τη συγκεκριμένη παρανόηση δεν υπάρχει κάποια σχετική αναφορά στη διεθνή βιβλιογραφία γιατί στα περισσότερα εισαγωγικά μαθήματα προγραμματισμού ακολουθείται διαφορετική προσέγγιση όσον αφορά τις δομές δεδομένων εννοιολογικά και φορμαλιστικά. Ωστόσο στο βιβλίο των Hazzan, Lapidot & Ragonis (2011), αναφέρεται πως όταν ένας αλγόριθμος χρησιμοποιεί ένα μικρό τμήμα ή ένα στοιχείο ενός πίνακα, οι μαθητές πιστεύουν ότι ο αλγόριθμος διατρέχει υποχρεωτικά όλα τα στοιχεία του πίνακα. Αντιλαμβάνονται δηλαδή τον πίνακα ως μια ενιαία οντότητα.

Η τελευταία παρανόηση που εντοπίσαμε σε αυτή την έρευνα και η οποία δεν αναφέρεται στη βιβλιογραφία, έχει να κάνει με την δυναμική σχέση μεταξύ του δείκτη i και του αντίστοιχου στοιχείου του πίνακα X[i] που βρίσκεται στην θέση i. Πρόκειται για μια συναρτησιακή σχέση μεταξύ του i και του X[i] που χαρακτηρίζεται από έναν δυναμικό χαρακτήρα, τον οποίο στην συγκεκριμένη έρευνα έδειξαν ότι αντιλαμβάνονται μόνο το 11% των μαθητών.

Στον Πίνακα 5.6 συνοψίζονται οι δυσκολίες και οι παρανοήσεις των μαθητών, που καταγράφηκαν στην έρευνα αυτή, ενώ επιχειρείται μια σύνδεση με αντίστοιχες νοητικές αναπαραστάσεις.

**Πίνακας 5.6. Λίστα παρανοήσεων των μαθητών στην έννοια του πίνακα**

| Δυσκολίες μαθητών | Παρανόηση | Αναπαράσταση |
|---|---|---|
| Να χρησιμοποιούν ένα στοιχείο πίνακα ως μεταβλητή | Δεν αντιλαμβάνονται ότι ένα στοιχείο πίνακα είναι ουσιαστικά μια μεταβλητή. | Ένα στοιχείο πίνακα δεν είναι μια απλή μεταβλητή. |
| Να διακρίνουν πότε απαιτείται η χρήση πίνακα για την επίλυση ενός προβλήματος | Το μοντέλο στοίβας/σωρού για τη μεταβλητή εμποδίζει αυτή τη διάκριση. | Μια μεταβλητή μπορεί να κρατήσει περισσότερες από μια τιμές άρα δεν χρειάζεται πίνακας |
| Η (ακέραια) τιμή ενός στοιχείου πίνακα δε μπορεί να χρησιμοποιηθεί ως δείκτης ενός άλλου στοιχείου πίνακα | Σύγχυση μεταξύ της τιμής ενός στοιχείου και της θέσης (δείκτη) του. | Ένα στοιχείο πίνακα και ο δείκτης σε αυτό μπορούν να αναπαρασταθούν ως μεταβλητές. |
| Οι μαθητές αναφέρονται σε όλο τον πίνακα με τον συμβολισμό X[i] | Το X[i] αναπαριστά όλο τον πίνακα και όχι ένα μόνο στοιχείο | Το X[i] αναπαριστά όλο τον πίνακα και όχι ένα μόνο στοιχείο |
| Το X[i] δεν αλλάζει όταν αλλάζει το i. (στατικό μοντέλο) | Το στοιχείο X[i] μένει το ίδιο μετά την ενημέρωση του i. | Στατικό μοντέλο του στοιχείου X[i] |

Το τελευταίο ερευνητικό ερώτημα της έρευνας είχε ως σκοπό να εξετάσει αν η ταξινομία SOLO θα μπορούσε να χρησιμοποιηθεί ως ένα αποτελεσματικό και αξιόπιστο πλαίσιο για την αξιολόγηση και ανάλυση των αναπαραστάσεων των μαθητών σχετικά με την έννοια του πίνακα. Η κατανόηση των προγραμματιστικών εννοιών από τους μαθητές προχωράει, συνήθως, από τα επιφανειακά σε πιο σταθερά και πληρέστερα νοητικά μοντέλα.

Η βαθμιαία ενίσχυση και ο εμπλουτισμός των νοητικών μοντέλων των μαθητών μπορεί να αντικατοπτριστεί στα πέντε επίπεδα αυξάνουσας γνωστικής πολυπλοκότητας της ταξινομίας SOLO. Η χρήση της ταξινομίας SOLO στην έρευνα αυτή επέτρεψε την ανάλυση των απαντήσεων των μαθητών και ανέδειξε, σε σημαντικό βαθμό, το επίπεδο αφαίρεσης και τις ποιοτικές διαφορές των προσεγγίσεων επίλυσης αλγοριθμικών προβλημάτων σε πίνακες.

Τα ευρήματα της έρευνας αυτής επιβεβαιώνουν προηγούμενες εμπειρικές μελέτες, οι οποίες έδειξαν ότι η ταξινομία SOLO προσφέρει ένα αποτελεσματικό πλαίσιο για την ανάλυση και την ερμηνεία των νοητικών μοντέλων των μαθητών για προγραμματιστικές έννοιες και δομές (Corney et al., 2014 · Lister et al., 2006 · 2009· Seiter, 2015· Sheard et al., 2008· Whalley, et al, 2011).

## *5.5 Συμπεράσματα*

Σχετικά με τις δυσκολίες των μαθητών στους πίνακες, υπάρχει στη διεθνή βιβλιογραφία μόνο η εργασία των Xinogalos, Satratzemi, & Dagdilelis (2008), η οποία όμως αναφέρεται στη δομή ArrayList της Java. Οι δυσκολίες των μαθητών στην δομή αυτή σχετίζονται περισσότερο με τις δυσκολίες του αντικειμενοστρεφούς προγραμματισμού και όχι με τις εγγενείς δυσκολίες της δομής του πίνακα στην αλγοριθμική επίλυση προβλημάτων.

Από τα αποτελέσματα της έρευνας προκύπτει ότι η πλειονότητα των μαθητών του δείγματος δεν έχουν οικοδομήσει λειτουργικές αναπαραστάσεις για την έννοια του πίνακα και δεν μπορούν να διακρίνουν την αναγκαιότητα χρήσης του σε αλγόριθμους επεξεργασίας δεδομένων του ίδιου τύπου. Ένα σημαντικό εύρημα είναι ότι κάποιες παρανοήσεις που έχουν οι μαθητές για την έννοια του πίνακα προέρχονται από την έννοια της μεταβλητής πάνω στην οποία οικοδομείται η έννοια του πίνακα.

Οι περισσότερες απαντήσεις των μαθητών κατατάχθηκαν μεταξύ του μονοδομικού και του πολυδομικού επιπέδου της ταξινομίας SOLO. Η κατάταξη των απαντήσεων

στα επίπεδα αυτά δείχνει ότι, ενώ οι μαθητές μπορούν να περιγράψουν τη λειτουργία κάθε μιας εντολής μεμονωμένα, δεν έχουν αναπτύξει ολοκληρωμένες αναπαραστάσεις για το ρόλο που επιτελούν συγκεκριμένα τμήματα αλγορίθμου.

Οι διδακτικές παρεμβάσεις στα σχετικά μαθήματα θα πρέπει να στοχεύουν στην ανάδειξη των βασικών χαρακτηριστικών της δομής του πίνακα, στην κατανόηση των σχετιζόμενων εννοιών και των δυναμικών χαρακτηριστικών τους, για παράδειγμα δείκτης, κελί, δεικτοδότηση, προσπέλαση ή καταχώρηση δεδομένων κ.λπ., καθώς και στην οικοδόμηση επαρκών αναπαραστάσεων για αλγορίθμους επεξεργασίας πινάκων.

# 6

# *Μελέτη των δυσκολιών των μαθητών στον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής*

## *6.1 Εισαγωγή*

Οι παρανοήσεις και δυσκολίες των μαθητών σε βασικές κατηγορίες αλγορίθμων, όπως είναι οι αλγόριθμοι ταξινόμησης ή αναζήτησης, δεν έχουν μελετηθεί εκτεταμένα (Geller & Dios, 1998· Simon et al., 2006). Η ταξινόμηση έχει περίοπτη θέση στα εισαγωγικά μαθήματα προγραμματισμού και αλγοριθμικής όχι μόνο γιατί αποτελεί ένα από τα θεμελιώδη αλγοριθμικά προβλήματα αλλά, κυρίως, γιατί είναι το συνηθέστερο παράδειγμα εισαγωγής των μαθητών και των φοιτητών σε τεχνικές σχεδίασης και ανάλυσης αλγορίθμων. Ο αλγόριθμος ταξινόμησης ευθείας ανταλλαγής βασίζεται στη σύγκριση και την αντιμετάθεση γειτονικών στοιχείων ενός πίνακα μέχρι τα στοιχεία του πίνακα να διαταχθούν στην επιθυμητή σειρά (αύξουσα/φθίνουσα). Σε πολλές περιπτώσεις, ο αλγόριθμος επιλέγεται για την μύηση των αρχάριων προγραμματιστών, κυρίως, λόγω της σύντομης κωδικοποίησής του.

```
Για i από 2 μέχρι N
    Για j από N μέχρι i με βήμα -1
        Αν A[ j ] < A[ j-1] Τότε
            Αντιμετάθεσε A[ j ], A[ j-1]
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 6.1. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής**

Όπως φαίνεται στο Τμήμα Κώδικα 6.1, ο αλγόριθμος μπορεί να διατυπωθεί με μόλις τρεις εντολές κώδικα, κάτι που διευκολύνει τη διαχείριση και τον γρήγορο προγραμματισμό του.

Αρκετοί επιστήμονες και εκπαιδευτικοί έχουν ασκήσει έντονη κριτική για την επιλογή της ευθείας ανταλλαγής ως εισαγωγικού αλγορίθμου ταξινόμησης αλλά και για την πρακτική χρησιμότητά της (Astrachan, 2003· Knuth 1998). Σε αντίθεση με τους αλγόριθμους επιλογής (selection sort) και εισαγωγής (insertion sort), οι δυσκολίες που αντιμετωπίζουν οι αρχάριοι προγραμματιστές οφείλονται στο γεγονός ότι ο αλγόριθμος της φυσαλίδας α) δεν προκύπτει άμεσα από την εμπειρία και β) δεν μπορεί να χτιστεί εύκολα πάνω στις προϋπάρχουσες γνώσεις τους (Geller & Dios, 1998· Hadas, 2013· Simon et al., 2006).

Αυτό επιβεβαιώνεται και από τις έρευνες που έχουν γίνει σε μαθητές δευτεροβάθμιας εκπαίδευσης στη χώρα μας για τη διερεύνηση των προϋπαρχουσών γνώσεων στους αλγόριθμους ταξινόμησης (Βραχνός & Τζιμογιάννης, 2018· 2016). Τα παραπάνω, σε συνδυασμό με τη διδακτική εμπειρία των ερευνητών σχετικά με τους αλγόριθμους ταξινόμησης, αποτέλεσαν το κίνητρο για την πραγματοποίηση αυτής της έρευνας.

## *6.2 Ερευνητικά ερωτήματα*

Ο βασικός στόχος της έρευνας ήταν να μελετήσουμε τις αντιλήψεις και τις αναπαραστάσεις μαθητών Λυκείου και φοιτητών Πληροφορικής σχετικά με τη λειτουργία του αλγορίθμου ταξινόμησης φυσαλίδας. Τα ερευνητικά ερωτήματα που τέθηκαν ήταν τα εξής:

1. Ποιες δυσκολίες και παρανοήσεις εμφανίζουν οι μαθητές-φοιτητές για τους αλγόριθμους ταξινόμησης;
2. Σε ποιο βαθμό οι μαθητές-φοιτητές διακρίνουν αν ένας αλγόριθμος εκτελεί ταξινόμηση;
3. Υπάρχουν σημαντικές διαφορές στις προσεγγίσεις μεταξύ μαθητών της Γ' τάξης του Λυκείου και πρωτοετών φοιτητών πληροφορικής;

## *6.3 Μεθοδολογία Έρευνας*

### *6.3.1 Πειραματικός σχεδιασμός*

Η έρευνα έλαβε χώρα τουλάχιστον ένα μήνα μετά τη διδασκαλία της σχετικής ενότητας που αναφέρεται στον αλγόριθμο ταξινόμησης φυσαλίδας. Δεν υπήρξε καμία διδακτική παρέμβαση πριν τη διεξαγωγή της έρευνας. Οι συμμετέχοντες μαθητές-φοιτητές είχαν διδαχθεί τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής από τους καθηγητές τους. Για τη συμπλήρωση του ερωτηματολογίου διατέθηκε μια διδακτική ώρα, στην οποία ήταν παρών ο ερευνητής και ο διδάσκων κάθε τάξης ή του μαθήματος προγραμματισμού στη γλώσσα C του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Πελοποννήσου.

### *6.3.2 Δείγμα*

Στην έρευνα συμμετείχαν 168 μαθητές της Γ' λυκείου Τεχνολογικής Κατεύθυνσης τριών γενικών λυκείων της ευρύτερης αστικής περιοχής της Αθήνας που παρακολουθούσαν το μάθημα Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον. Επίσης συμμετείχαν 53 πρωτοετείς φοιτητές του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Πελοποννήσου, οι οποίοι είχαν ολοκληρώσει το εισαγωγικό μάθημα προγραμματισμού στη γλώσσα C του 1ου εξαμήνου. Η πλειονότητα των μαθητών και των φοιτητών δήλωσαν ότι έχουν πρόσβαση σε υπολογιστή στο σπίτι, και ότι έχουν αναπτύξει τουλάχιστον ένα πρόγραμμα σε κάποια γλώσσα προγραμματισμού.

Ωστόσο μετά την ανάλυση των απαντήσεων του πρώτου έργου αφαιρέσαμε από το δείγμα, 36 μαθητές και 3 φοιτητές οι οποίοι έδειξαν πως δεν μπορούν να εφαρμόσουν με χαρτί και μολύβι τα βήματα του αλγόριθμου ταξινόμησης της ευθείας ανταλλαγής ώστε να τοποθετήσουν σε αύξουσα σειρά τα στοιχεία ενός μικρού πίνακα αριθμών. Οι παραπάνω μαθητές και φοιτητές φάνηκε ότι δεν μπορούσαν να απαντήσουν σε απλά ερωτήματα όπως τι είναι ταξινόμηση, αύξουσα και φθίνουσα διάταξη αριθμών.

### *6.3.3 Εργαλείο της έρευνας*

Για την καταγραφή των αντιλήψεων και των αναπαραστάσεων των μαθητών σχετικά με τον αλγόριθμο ταξινόμησης φυσαλίδας χρησιμοποιήθηκε ερωτηματολόγιο, το οποίο σχεδιάστηκε και αναπτύχθηκε ειδικά για την παρούσα μελέτη. Το

ερωτηματολόγιο περιλάμβανε πέντε αλγοριθμικά προβλήματα που σχεδιάστηκαν με βάση τους παρακάτω άξονες:

**α) Ανάγνωση Κώδικα:** Σε ποιο βαθμό μπορεί ο μαθητής/φοιτητής να αναγνωρίσει τη λειτουργία που επιτελεί ένα τμήμα αλγορίθμου; (π.χ. αύξουσα/φθίνουσα ταξινόμηση, εύρεση μεγίστου/ελαχίστου κλπ.)

**β) Αναγκαιότητα του αλγορίθμου ταξινόμησης:** Κατά πόσο οι μαθητές/φοιτητές είναι σε θέση να διακρίνουν την αναγκαιότητα του αλγορίθμου ταξινόμησης σε ένα πρόβλημα επεξεργασίας δεδομένων;

**γ) Ευστάθεια αλγορίθμου ταξινόμησης:** Ένας αλγόριθμος ταξινόμησης είναι ευσταθής όταν τα στοιχεία με την ίδια τιμή εμφανίζονται στην έξοδο με την ίδια σειρά που εμφανίζονται στην είσοδο. Για παράδειγμα, ο αλγόριθμος της φυσαλίδας είναι ευσταθής ενώ ο αλγόριθμος της γρήγορης ταξινόμησης (quick sort) όχι. Μπορούν οι μαθητές να διακρίνουν τη ευστάθεια ενός αλγορίθμου ταξινόμησης;

## *6.4 Ανάλυση των αποτελεσμάτων*

Οι απαντήσεις των μαθητών και των φοιτητών αναλύθηκαν με βάση την ταξινομία SOLO. Στη συνέχεια, παρουσιάζονται τα αποτελέσματα από τα πέντε έργα του ερωτηματολογίου.

### *6.4.1 Έργο 1*

*Έστω ο πίνακας Α με στοιχεία όπως παρακάτω:*

| | | | | | |
|---|---|---|---|---|---|
| 40 | 35 | 25 | 15 | 10 | 5 |

*Στον πίνακα Α εκτελείται ο αλγόριθμος ταξινόμησης της φυσαλίδας. Θεωρείστε ότι, για κάποιον απρόβλεπτο (τεχνικό) λόγο, ο αλγόριθμος σταματάει χωρίς να έχει ολοκληρωθεί η ταξινόμηση. Έστω ότι τη στιγμή της διακοπής ο πίνακας έχει τη μορφή:*

| | | | | | |
|---|---|---|---|---|---|
| 5 | 10 | 15 | 40 | 25 | 35 |

*α) Η ταξινόμηση είναι αύξουσα ή φθίνουσα; Να αιτιολογήσετε την απάντησή σας.*

*β) Πόσα στοιχεία του πίνακα έχουν ήδη ταξινομηθεί (δηλαδή βρίσκονται στις τελικές θέσεις ταξινόμησης σύμφωνα με τον αλγόριθμο φυσαλίδας);*

*γ) Ποια είναι τα επόμενα στοιχεία που θα συγκριθούν, όταν η ταξινόμηση ξεκινήσει ξανά από το σημείο διακοπής;*

*δ) Ποια είναι τα επόμενα στοιχεία που θα αλλάξουν θέση, όταν η ταξινόμηση ξεκινήσει ξανά από το σημείο διακοπής;*

Το πρώτο έργο, είχε διττό σκοπό. Ο πρώτος στόχος ήταν να εξετάσουμε αν υπάρχουν μαθητές οι οποίοι δεν γνωρίζουν τη λειτουργία του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής ώστε να μπορούν να τον εφαρμόσουν σε μια σειρά από αντικείμενα. Ο δεύτερος στόχος ήταν να αποκτήσουν αυτοπεποίθηση οι μαθητές μέσα από τη λύση ενός απλού προβλήματος στην αρχή ώστε να μπορέσουν να απαντήσουν σε όσα περισσότερα προβλήματα μπορούν, στη συνέχεια.

Η πλειονότητα των μαθητών και των φοιτητών απάντησε σωστά στα πρώτα ερωτήματα και έδειξαν ότι μπορούν να ταξινομήσουν μια σειρά από αντικείμενα σύμφωνα με τη λογική του αλγορίθμου της ευθείας ανταλλαγής, εκτός από 32 μαθητές και 3 φοιτητές των οποίων οι απαντήσεις δεν λήφθηκαν υπόψη στη συνέχεια.

### *6.4.2 Έργο 2*

*Δίνεται ο πίνακας Α με Ν ακέραιους αριθμούς και ο παρακάτω αλγόριθμος:*

**Για** i **από** N **μέχρι** 2 **με βήμα** –1
  **Αν** A[ i ] < A[ i–1 ] **Τότε**
    temp ← A[ i ]
    A[ i ] ← A[ i–1 ]
    A[ i–1 ] ← temp
  **Τέλος_Αν**
**Τέλος_Επανάληψης**

*Τι αναμένετε να συμβεί κατά την εκτέλεση του αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του.*

Το παραπάνω Τμήμα Κώδικα εκτελεί ένα μόνο πέρασμα του αλγορίθμου ταξινόμησης της φυσαλίδας, το οποίο έχει ως αποτέλεσμα να "ανέβει" στην πρώτη θέση του πίνακα το ελάχιστο στοιχείο. Η απάντηση αυτή δόθηκε από 34 μαθητές και 18 φοιτητές και κατατάσσεται στο συνθετικό επίπεδο αφού αποτελεί περιγραφή της λειτουργίας του αλγορίθμου σε ένα αφηρημένο επίπεδο.

**Πίνακας 6.1. Απαντήσεις των μαθητών στο έργο 2**

| Επίπεδο SOLO | Μαθητές (N=136) | Φοιτητές (N=50) |
|---|---|---|
| | Ποσοστό % | Ποσοστό % |
| Προδομικό | 51.5 | 40.0 |
| Πολυδομικό | 23.5 | 24.0 |
| Συνθετικό | 25.0 | 36.0 |

Σχεδόν οι μισοί μαθητές και το 40% των φοιτητών απάντησαν ότι ο αλγόριθμος εκτελεί ταξινόμηση όλου του πίνακα. Οι απαντήσεις αυτές κατατάχθηκαν στο προδομικό επίπεδο αφού οι μαθητές δεν διέκριναν ότι πρόκειται για ένα μόνο πέρασμα του αλγορίθμου, κάτι που δείχνει ότι δεν μπόρεσαν να διακρίνουν τη βασική δομή ενός αλγόριθμου ταξινόμησης.

Οι υπόλοιπες απαντήσεις κατατάχθηκαν στο πολυδομικό επίπεδο αφού οι μαθητές έδωσαν αναλυτικές περιγραφές για κάθε εντολή, χωρίς όμως να διακρίνουν τη λειτουργία που επιτελεί ο αλγόριθμος. Μια χαρακτηριστική απάντηση αυτής της κατηγορίας δίνεται παρακάτω:

*«Για κάθε στοιχεία A[ i ], A[ i–1 ] αν το A[ i ] είναι μικρότερο από το A[ i–1 ] τότε τα στοιχεία A[ i ], A[ i–1 ] αλλάζουν θέση μεταξύ τους»*

Στην παραπάνω απάντηση φαίνεται ότι οι μαθητές έχουν κατανοήσει ότι το μπλοκ των τριών εκχωρήσεων εκτελεί αντιμετάθεση των τιμών. Αυτό συνιστά σκέψη που εντάσσεται σε ένα επίπεδο ανώτερο του πολυδομικού (Corney, Lister, & Teague, 2011). Ωστόσο στο πλαίσιο της έρευνας αυτές οι απαντήσεις κατατάχθηκαν στο πολυδομικό επίπεδο διότι οι περισσότεροι μαθητές και φοιτητές έχουν εξοικειωθεί με την συγκεκριμένη τριάδα εντολών και πλέον αναγνωρίζουν αμέσως τη λειτουργία που επιτελεί.

### *6.4.3 Έργο 3*

*Δίνεται ο πίνακας Α με Ν ακέραιους αριθμούς και ο παρακάτω αλγόριθμος:*

```
Για i από N-1 μέχρι 1 με βήμα –1
  Για j από 1 μέχρι i
    Αν A[j] < A[j+1] Τότε
      temp ← A[j]
      A[j] ← A[j+1]
      A[j+1] ← temp
    Τέλος_Αν
  Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

*Τι αναμένετε να συμβεί κατά την εκτέλεση του αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του.*

Η εκδοχή αυτή του αλγορίθμου είναι γνωστή στη βιβλιογραφία ως ταξινόμηση βυθού ή βαριδιού (sinking sort), καθώς τα στοιχεία του πίνακα αντί να κατευθύνονται προς τα πάνω σαν φυσαλίδες κινούνται προς την αντίθετη κατεύθυνση. Αυτό φαίνεται κυρίως από την εσωτερική επανάληψη που ξεκινά τις συγκρίσεις από το πρώτο στοιχείο του πίνακα. Μια άλλη δυσκολία που αναμένεται να συναντήσουν οι μαθητές είναι ότι αντί των 'γνώριμων', από το σχολικό εγχειρίδιο, στοιχείων A[j], A[j–1] εδώ συγκρίνονται τα A[j], A[j+1]. Η αλλαγή αυτή έγινε με στόχο να διερευνηθεί ο βαθμός κατανόησης του ρόλου των δεικτών i, j στον αλγόριθμο ταξινόμησης.

**Πίνακας 6.2. Απαντήσεις των μαθητών στο έργο 3**

| **Επίπεδο SOLO** | **Μαθητές (N=136)** | **Φοιτητές (N=50)** |
|---|---|---|
| | **Ποσοστό %** | **Ποσοστό %** |
| Δεν απάντησαν | 11.8 | 10.0 |
| Προδομικό | 16.2 | 8.0 |
| Πολυδομικό | 14.6 | 12.0 |
| Συνθετικό | 57.4 | 70.0 |

Συνολικά το 57.4% των μαθητών και το 70% των φοιτητών διέκριναν ότι ο παραπάνω αλγόριθμος εκτελεί ταξινόμηση. Οι απαντήσεις αυτές κατατάσσονται στο

συνθετικό επίπεδο της ταξινομίας SOLO. Ωστόσο αρκετοί μαθητές (37%) αλλά και φοιτητές (34%) ενώ αναγνώρισαν ότι ο αλγόριθμος εκτελεί ταξινόμηση στον πίνακα, δεν διέκριναν το είδος της και απάντησαν ότι είναι αύξουσα ταξινόμηση. Αυτό φαίνεται να εδράζεται στην αλλαγή της έκφρασης–δείκτη από j–1 σε j+1. Ωστόσο και αυτές οι απαντήσεις κατατάσσονται στο συνθετικό επίπεδο γιατί οι μαθητές διέκριναν τη λειτουργία του σε ένα αφηρημένο επίπεδο.

Αυτό που παρουσιάζει εξαιρετικό ενδιαφέρον, είναι ότι το 16.2% των μαθητών και το 8% των φοιτητών έδωσαν απαντήσεις οι οποίες κατατάχθηκαν στο προδομικό επίπεδο όπως:

*«Ο αλγόριθμος είναι λάθος.»*

«*Αντί για j+1 θέλει j–1 .*»

«*Ο αλγόριθμος δεν εκτελεί ταξινόμηση .*»

Αυτό αποτελεί ένδειξη ότι οι μαθητές έχουν απομνημονεύσει τον αλγόριθμο της ταξινόμησης ευθείας ανταλλαγής, όπως είναι διατυπωμένος στο σχολικό εγχειρίδιο, και δεν αναγνωρίζουν κάποια παραλλαγή του όταν την δουν, ούτε είναι σε θέση να τον τροποποιήσουν για κάποια ειδική περίπτωση.

Τελικά το 20% των μαθητών και το 36% των φοιτητών διέκριναν σωστά και το είδος της ταξινόμησης (φθίνουσα).

### *6.4.4 Έργο 4*

*Έστω ο πίνακας Α με στοιχεία όπως παρακάτω:*

| | | | | | |
|---|---|---|---|---|---|
| 39 | 23 | 29 | 14 | 38 | 23 |

*Ποια θα είναι η μορφή του πίνακα Α, μετά την εκτέλεση του παρακάτω αλγορίθμου;*

```
Για i από 4 μέχρι 6
    Για j από 6 μέχρι i με βήμα –1
        Αν A[j] < A[j–1] Τότε
            temp ← A[j]
            A[j] ← A[j–1]
            A[j–1] ← temp
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

*Να εξηγήσετε συνοπτικά τη λειτουργία του παραπάνω αλγορίθμου.*

Στο έργο αυτό δόθηκε στους μαθητές ένα τμήμα αλγορίθμου το οποίο εκτελεί ‘μερική’ ταξινόμηση του πίνακα Α, αφού εκτελεί μόνο 3 περάσματα του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής, άρα ταξινομεί 3 από τα 6 στοιχεία του πίνακα. Δεδομένου ότι η έκδοση του αλγορίθμου που δόθηκε στους μαθητές συμφωνεί με αυτή του σχολικού βιβλίου, η απάντηση ότι ο αλγόριθμος εκτελεί ταξινόμηση κατατάσσεται στο πολυδομικό επίπεδο της ταξινομίας SOLO και όχι στο συνθετικό, διότι οι μαθητές κάνουν απλά ανάκληση της μορφής του αλγορίθμου που έχουν απομνημονεύσει. Αντίθετα, όσες απαντήσεις επικεντρώθηκαν στη μερική ταξινόμηση ενός τμήματος του πίνακα, ακόμα αν και ήταν λανθασμένες όσον αφορά στο πλήθος ή στη διάταξη της ταξινόμησης, κατατάχθηκαν στο συνθετικό επίπεδο, καθώς θεωρούμε ότι έχουν κατανοήσει ότι η εξωτερική επανάληψη του αλγορίθμου καθορίζει πόσα περάσματα θα γίνουν και, επομένως, πόσα στοιχεία θα ταξινομηθούν.

**Πίνακας 6.3. Απαντήσεις των μαθητών στο έργο 4**

| **Επίπεδο SOLO** | **Μαθητές (N=136)** | **Φοιτητές (N=50)** |
|---|---|---|
| | **Ποσοστό %** | **Ποσοστό %** |
| Δεν απάντησαν | 20.6 | 34.0 |
| Πολυδομικό | 33.1 | 26.0 |
| Συνθετικό | 46.3 | 28.0 |

Είναι ιδιαίτερα ενδιαφέρον το γεγονός ότι οι μαθητές ανταποκρίθηκαν καλύτερα από τους πρωτοετείς φοιτητές του Τμήματος Πληροφορικής στο ερώτημα αυτό. Όπως φαίνεται στον Πίνακα 6.3, στο συνθετικό επίπεδο κατατάσσεται το 46,3% των απαντήσεων των μαθητών και μόνο το 28% των απαντήσεων των φοιτητών. Μια ερμηνεία θα μπορούσε να είναι κάποιοι μαθητές έχουν εμβαθύνει περισσότερο στον αλγόριθμο ταξινόμησης φυσαλίδας, καθώς είναι ο μοναδικός αλγόριθμος ταξινόμησης που διδάσκεται στο Λύκειο ενώ περιλαμβάνεται στην εξεταστέα ύλη των πανελλαδικών εξετάσεων.

Κάποιες από τις απαντήσεις που κατατάχθηκαν στο συνθετικό επίπεδο δεν ήταν σωστές, όμως έδειξαν ότι οι μαθητές έχουν αναγνωρίσει την λειτουργία του αλγορίθμου στο πιο αφηρημένο επίπεδο. Κάποιες χαρακτηριστικές απαντήσεις που κατατάχθηκαν στο συνθετικό επίπεδο δίνονται παρακάτω:

*«ταξινόμηση ενός τμήματος του πίνακα»*

*«ταξινόμηση των τριών τελευταίων στοιχείων»*

*«ταξινόμηση των τεσσάρων τελευταίων στοιχείων»*

Η πρώτη από τις απαντήσεις θα μπορούσε να θεωρηθεί ότι βρίσκεται μεταξύ πολυδομικού και συνθετικού επιπέδου.

### *6.4.5 Έργο 5*

*Δίνεται ο παρακάτω αλγόριθμος και οι πίνακες Όνομα και Βαθμός που αφορούν τα ονόματα και τις βαθμολογίες 5 μαθητών.*

```
Για i από 2 μέχρι 5
    Για j από 5 μέχρι i με βήμα –1
        Αν Βαθμός[j] > Βαθμός[j–1] Τότε
            Αντιμετάθεσε Βαθμός[j], Βαθμός[j–1]
            Αντιμετάθεσε Όνομα[j], Όνομα[j–1]
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

| | Όνομα | Βαθμός |
|---|---|---|
| 1 | Θανάσης | 18 |
| 2 | Μυρτώ | 19 |
| 3 | Ελένη | 18 |
| 4 | Μαρία | 20 |
| 5 | Γεωργία | 18 |

***α)*** *Ποια θα είναι τα πρώτα τέσσερα ονόματα μετά την εκτέλεση του αλγορίθμου;*

***β)*** *Τί αναμένετε να συμβεί στους 2 πίνακες, μετά την εκτέλεση του νέου αλγορίθμου, αν η συνθήκη Βαθμός[j] > Βαθμός[j–1] αλλάξει σε Βαθμός[j] >= Βαθμός[j–1];*

*Να αιτιολογήσετε την απάντησή σας.*

Το ενδιαφέρον του προβλήματος αυτού έγκειται στο γεγονός ότι για τη σύγκριση χρησιμοποιείται ένα χαρακτηριστικό των δεδομένων (βαθμός) ενώ αντιμετατίθενται τα στοιχεία του πίνακα Όνομα. Ο βαθμός σε αυτή την περίπτωση παίζει το ρόλο του κλειδιού που καθορίζει τη διάταξη των μαθητών. Το δεύτερο ερώτημα αλλάζει την ευστάθεια του αλγορίθμου.

Το 61% των μαθητών και το 44% των φοιτητών δεν απάντησαν ή οι απαντήσεις τους δεν ήταν σχετικές με το πλαίσιο του προβλήματος. Το 10.3% των μαθητών απάντησε ότι στις ισοβαθμίες η ταξινόμηση γίνεται αλφαβητικά. Στις απαντήσεις που έδωσαν η σειρά της Γεωργίας, της Ελένης και του Θανάση άλλαξε παρόλο που και οι τρεις έχουν τον ίδιο βαθμό.

Μια εξήγηση για αυτή την προσέγγιση είναι ότι οι μαθητές προσπάθησαν να ανακαλέσουν ένα γνωστό πρόβλημα που να μοιάζει με αυτό που τους δόθηκε. Στο λύκειο οι μαθητές αντιμετωπίζουν πολλές παρόμοιες ασκήσεις στις οποίες κατά την ισοβαθμία τα ονόματα διατάσσονται αλφαβητικά. Έτσι αντί να προσαρμόσουν τη λύση που γνωρίζουν στο πρόβλημα που τους δόθηκε, προσάρμοσαν το πρόβλημα ώστε να ταιριάζει στη λύση που τους είναι οικεία.

**Πίνακας 6.4. Απαντήσεις των μαθητών στο έργο 5**

| **Επίπεδο SOLO** | **Μαθητές (N=136)** | **Φοιτητές (N=50)** |
|---|---|---|
| | **Ποσοστό %** | **Ποσοστό %** |
| Δεν απάντησαν | 41.2 | 36.0 |
| Προδομικό | 33.0 | 18.0 |
| Πολυδομικό | 14.0 | 32.0 |
| Συνθετικό | 11.8 | 14.0 |

Τελικά μόνο το 11.8% των μαθητών και το 14% φοιτητών διέκρινε ότι στην ισοβαθμία η σειρά θα αλλάξει αλλά όχι αλφαβητικά και για αυτό οι απαντήσεις τους κατατάχθηκαν στο συνθετικό επίπεδο. Μάλιστα παρουσιάζουν ενδιαφέρον οι απαντήσεις μαθητών όπως:

*«Θα έχουμε συνεχείς εναλλαγές επ' άπειρον.»*

*«Ο αλγόριθμος δεν θα σταματήσει ποτέ.»*

*«Οι αντιμεταθέσεις των στοιχείων δεν σταματούν.»*

Οι μαθητές δεν διέκριναν ότι ο αλγόριθμος εκτελείται για συγκεκριμένο πλήθος επαναλήψεων, αλλά κατασκεύασαν στο μυαλό τους ένα νοητό μοντέλο ταξινόμησης, στο οποίο ο αλγόριθμος τερματίζει μόνο όταν σταματήσουν οι αντιμεταθέσεις των στοιχείων του πίνακα.

Άλλοι μαθητές απάντησαν ότι

«*θα υπάρξει πρόβλημα με όσους έχουν ίδιους βαθμούς και θα μπερδευτεί ο αλγόριθμος*»

Μια άλλη ενδιαφέρουσα απάντηση ήταν ότι

«*ο πίνακας θα έχει μια λιγότερη τιμή αφού οι βαθμοί του Θανάση και της Γεωργίας θα καταλάβουν την ίδια θέση στον πίνακα*»

ή

«*τα ονόματα θα μπουν στο ίδιο κουτάκι*»

Η παρανόηση που έχουν οι μαθητές σε αυτή την περίπτωση έχει τις ρίζες της σε μια άλλη παρανόηση για την έννοια της μεταβλητής, πάνω στην οποία χτίζεται η έννοια του πίνακα. Στην συγκεκριμένη περίπτωση οι μαθητές έχουν την παρανόηση ότι μια μεταβλητή μπορεί να διατηρήσει περισσότερες από μια τιμές και σε κάποιες περιπτώσεις ότι διατηρεί όλο το ιστορικό των τιμών που έχει λάβει (Jimoyiannis, 2011). Το εύρημα αυτό επιβεβαιώνει τα αποτελέσματα παλαιότερης έρευνας σχετικά με τις παρανοήσεις και τις δυσκολίες των μαθητών σε προβλήματα με πίνακες (Βραχνός και Τζιμογιάννης, 2010).

### *6.4.6 Έργο 6*

*Έστω ο πίνακας **A** με στοιχεία όπως παρακάτω:*

| | | | | | |
|---|---|---|---|---|---|
| 15 | 10 | 15 | 12 | 1 | 8 |

*Ποια θα είναι η μορφή του πίνακα Α, μετά την εκτέλεση του διπλανού τμήματος κώδικα; Να εξηγήσετε συνοπτικά τη λειτουργία του αλγορίθμου.*

```
Για i από 2 μέχρι 6
    Για j από 6 μέχρι i με βήμα –1
        Αν A[j] mod 2 < A[j–1] mod 2 Τότε
            temp ← A[j]
            A[j] ← A[j–1]
            A[j–1] ← temp
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

*Υπενθυμίζεται ότι η έκφραση **α mod 2**, υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του αριθμού α με το 2. Για παράδειγμα, 4 mod 2 = 0, 3 mod 2 = 1, 1 mod 2 = 1.*

Ο αλγόριθμος ταξινομεί τα στοιχεία του πίνακα σε αύξουσα σειρά ως προς το υπόλοιπο της διαίρεσης τους με το 2, διαχωρίζοντας τους άρτιους από τους περιττούς αριθμούς, χωρίς όμως να αλλάζει η σχετική σειρά μεταξύ τους. Οι άρτιοι αριθμοί τοποθετούνται πριν τους περιττούς.

Δεκαεπτά (17) μαθητές και εννιά (9) φοιτητές έδωσαν απαντήσεις που κατατάχθηκαν στο συνθετικό επίπεδο όπως αυτές που ακολουθούν:

*«ο αλγόριθμος διαχωρίζει τους ζυγούς από τους μονούς αριθμούς»*

*«Οι άρτιοι αριθμοί πηγαίνουν στις πρώτες θέσεις του πίνακα και οι περιττοί στις τελευταίες»*

Κανένας δεν παρατήρησε ότι η σχετική σειρά των άρτιων και των περιττών μεταξύ τους δεν αλλάζει.

Ερμηνείες που κατατάχθηκαν στο πολυδομικό επίπεδο έδωσαν 16 φοιτητές και 34 μαθητές. Σε αυτές τις απαντήσεις δόθηκαν περιγραφές του κώδικα όπως η παρακάτω:

*«ο αλγόριθμος συγκρίνει τα υπόλοιπα των στοιχείων του πίνακα Α με το 2 και ταξινομεί τα στοιχεία κατά αύξουσα σειρά των υπολοίπων.»*

Η παραπάνω απάντηση δείχνει ότι ο μαθητής έχει κατανοήσει τη λειτουργία όλων των εντολών αλλά δεν μπορεί να συνδέσει τα επιμέρους τμήματα του αλγορίθμου ώστε να διακρίνει την συνολική λειτουργία του.

Τέλος, αρκετοί μαθητές (40.5%) και φοιτητές (34%) απάντησαν ότι ο αλγόριθμος εκτελεί ταξινόμηση των αριθμών του πίνακα, είτε αύξουσα είτε φθίνουσα. Εδώ φαίνεται ότι οι μαθητές δεν μπορούν να διαχωρίσουν την έννοια του χαρακτηριστικού (κλειδί) βάσει του οποίου ορίζεται η διάταξη των στοιχείων του πίνακα και των ίδιων των στοιχείων του πίνακα, γιατί έχουν εξοικειωθεί με την ιδέα ότι πάντα συγκρίνονται γειτονικά στοιχεία του πίνακα.

**Πίνακας 6.5. Απαντήσεις των μαθητών στο έργο 6**

| **Επίπεδο SOLO** | **Μαθητές (N=136)** | **Φοιτητές (N=50)** |
|---|---|---|
| | **Ποσοστό %** | **Ποσοστό %** |
| Δεν απάντησαν | 22.0 | 16.0 |
| Προδομικό | 40.5 | 34.0 |
| Πολυδομικό | 25.0 | 32.0 |
| Συνθετικό | 12.5 | 18.0 |

## *Συζήτηση*

Από τα αποτελέσματα της έρευνας προκύπτει ότι η πλειονότητα των μαθητών και των φοιτητών του δείγματος δεν έχουν οικοδομήσει λειτουργικές αναπαραστάσεις για την έννοια του αλγορίθμου ταξινόμησης και δεν μπορούν να διακρίνουν αν ένας αλγόριθμος εκτελεί ταξινόμηση ή όχι. Οι μαθητές αναγνωρίζουν μόνο τον συγκεκριμένο αλγόριθμο ταξινόμησης ευθείας ανταλλαγής όπως αυτός ακριβώς παρουσιάζεται στο σχολικό βιβλίο. Οποιαδήποτε παραλλαγή, όσο μικρή και αν είναι, δημιουργεί σύγχυση και εισάγει δυσκολίες. Στα έργα με παραλλαγές του αλγορίθμου ευθείας ανταλλαγής λίγοι μαθητές (κάτω από 20%) κατάφεραν να δώσουν απαντήσεις που κατατάχθηκαν στο συνθετικό επίπεδο της ταξινομίας SOLO. Οι περισσότερες απαντήσεις των μαθητών κατατάχθηκαν μεταξύ του μονοδομικού και του πολυδομικού επιπέδου της ταξινομίας. Η κατάταξη των απαντήσεων στα επίπεδα αυτά δείχνει ότι, ενώ οι μαθητές μπορούν να περιγράψουν τη λειτουργία κάθε μιας εντολής μεμονωμένα, δεν έχουν αναπτύξει ολοκληρωμένες αναπαραστάσεις για το ρόλο που επιτελούν συγκεκριμένα τμήματα του αλγορίθμου. Σε ένα πρόβλημα οι μαθητές έδωσαν περισσότερες απαντήσεις στο συνθετικό επίπεδο της ταξινομίας SOLO από ότι οι φοιτητές. Αυτό ενδεχομένως οφείλεται στο ότι η μορφή του αλγόριθμου ταξινόμησης ήταν αρκετά κοντά σε αυτή που γνωρίζουν οι μαθητές από το σχολικό βιβλίο. Στα υπόλοιπα προβλήματα, τα οποία ήταν περισσότερο αυθεντικά, οι φοιτητές είχαν καλύτερες επιδόσεις από τους μαθητές.

Μια εξήγηση για τις δυσκολίες που παρουσιάζουν οι μαθητές και οι φοιτητές είναι ότι ο αλγόριθμος της ευθείας ανταλλαγής είναι δύσκολο να χτιστεί σε προϋπάρχουσα γνώση σε αντίθεση με τους αλγόριθμους επιλογής και εισαγωγής (Geller & Dios, 1998· Hadas, 2013· Simon et al., 2006), για τους οποίους είναι πιο εύκολο για τον εκπαιδευτικό να χτίσει σκαλωσιές μάθησης. Για παράδειγμα, ο αλγόριθμος ταξινόμησης επιλογής μπορεί να οικοδομηθεί πάνω στην εύρεση του ελάχιστου στοιχείου πίνακα, κάτι που οι μαθητές ήδη γνωρίζουν. Επίσης, επειδή οι αλγόριθμοι ταξινόμησης χτίζονται πάνω στις έννοιες της μεταβλητής, του πίνακα και των δομών επανάληψης, είναι αναμενόμενο οι μαθητές να ‘κληρονομούν’ πολλές από τις παρανοήσεις για αυτές τις έννοιες (Vrachnos & Jimoyiannis, 2017).

## *6.5 Συμπεράσματα*

Από τα αποτελέσματα της έρευνας που έγινε προκύπτει ότι οι μαθητές και οι φοιτητές συνάντησαν μεγάλες δυσκολίες στην αντιμετώπιση προβλημάτων που είχαν να κάνουν με τη λειτουργία του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής. Αυτό συμβαίνει επειδή έχουν μάθει να αναγνωρίζουν και να χρησιμοποιούν μια συγκεκριμένη εκδοχή του αλγορίθμου με αποτέλεσμα να αντιμετωπίζουν μεγάλες δυσκολίες όταν καλούνται να μελετήσουν μικρές παραλλαγές του.

Οι περισσότερες απαντήσεις των μαθητών και των φοιτητών σε αυτή την έρευνα κατατάχθηκαν μεταξύ του μονοδομικού και του πολυδομικού επιπέδου της ταξινομίας SOLO. Η κατάταξη των απαντήσεων στα επίπεδα αυτά δείχνει ότι, ενώ οι μαθητές και οι φοιτητές μπορούν να περιγράψουν τη λειτουργία κάθε μιας εντολής μεμονωμένα, δεν έχουν αναπτύξει ολοκληρωμένες αναπαραστάσεις για το ρόλο που επιτελούν συγκεκριμένα τμήματα αλγορίθμου.

Ο σχεδιασμός κατάλληλων διδακτικών παρεμβάσεων πρέπει να στοχεύει στην ανάδειξη των βασικών χαρακτηριστικών της δομής του πίνακα και των βασικών λειτουργιών του αλγορίθμου ταξινόμησης, όπως είναι η αντιμετάθεση, η σύγκριση και η προσπέλαση. Οι νέοι αλγόριθμοι, με τους οποίους έρχονται σε επαφή για πρώτη φορά οι αρχάριοι προγραμματιστές (μαθητές-φοιτητές), πρέπει να οικοδομηθούν πάνω στις προϋπάρχουσες γνώσεις και εμπειρίες τους (Hadas, 2013). Οι επαρκείς αναπαραστάσεις δεν είναι εύκολο να οικοδομηθούν ή να παρουσιαστούν με την παραδοσιακή διδασκαλία στον πίνακα ή/και με παραδείγματα που υλοποιούν οι μαθητές με χαρτί και μολύβι. Η χρήση εναλλακτικών περιβαλλόντων, όπως είναι τα περιβάλλοντα δυναμικής οπτικοποίησης αλγορίθμων (Liu et al., 2009· Naps et al., 2000· Urquiza–Fuentes & Velazquez–Iturbide, 2009· Vrachnos & Jimoyiannis, 2014), παρέχουν στους μαθητές δυνατότητες για πειραματισμό και διερεύνηση και μπορούν να συμβάλλουν στην οικοδόμηση επαρκών αναπαραστάσεων για τις δομές δεδομένων και τους αλγορίθμους ταξινόμησης.

# 7

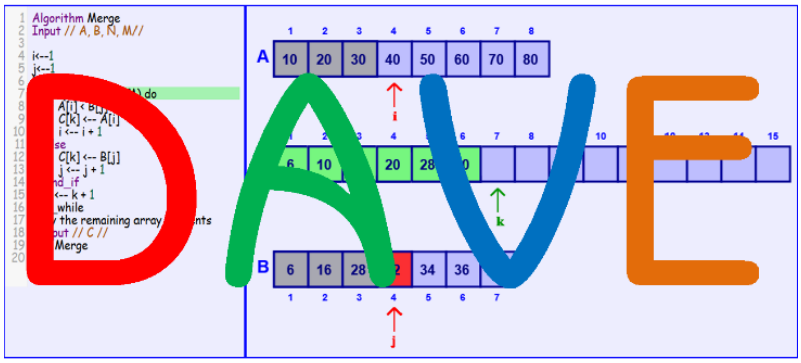
# *Σχεδιασμός του Περιβάλλοντος Οπτικοποίησης Αλγορίθμων DAVE*

## *7.1 Εισαγωγή*

Στο προηγούμενο κεφάλαιο παρουσιάσαμε τις δυσκολίες των μαθητών και των φοιτητών στους πίνακες και στους αλγόριθμους ταξινόμησης. Οι έννοιες αυτές είναι δύσκολο να οικοδομηθούν με τα παραδοσιακά διδακτικά μέσα λόγω της δυναμικής φύσης τους. Ένας αλγόριθμος ταξινόμησης ο οποίος βασίζεται σε συνεχείς συγκρίσεις και αντιμεταθέσεις στοιχείων δεν είναι εύκολο να παρουσιαστεί στους μαθητές με συμβατικά μέσα.

Τα συνήθη προγραμματιστικά περιβάλλοντα και οι γλώσσες προγραμματισμού που χρησιμοποιούνται για εκπαιδευτικούς σκοπούς έχουν σχεδιαστεί για την ανάπτυξη εφαρμογών και όχι για τη διδασκαλία του προγραμματισμού. Είναι, συνεπώς, προσαρμοσμένα στο πλαίσιο γνώσεων και δεξιοτήτων των έμπειρων προγραμματιστών, γεγονός που ενισχύει τις δυσκολίες και τα εμπόδια που συναντούν οι μαθητές και οι αρχάριοι στον προγραμματισμό (Κόμης & Τζιμογιάννης, 2006).

Οι παραπάνω διαπιστώσεις αποτέλεσαν το κίνητρο για τον σχεδιασμό και την υλοποίηση του εκπαιδευτικού περιβάλλοντος οπτικοποίησης αλγορίθμων DAVE (**D**ynamic **A**lgorithm **V**isualization **E**nvironment). Ο σκοπός του συστήματος είναι να διευκολύνει τους μαθητές στην κατανόηση της λειτουργίας των αλγορίθμων μέσα από τον πειραματισμό και την διερεύνηση της εκτέλεσής τους με διάφορες κατηγορίες δεδομένων.




**Σχήμα 7.1. Ο λογότυπος του συστήματος DAVE**

## *7.2 Χαρακτηριστικά του περιβάλλοντος DAVE*

Η επισκόπηση των συστημάτων οπτικοποίησης αλγορίθμων που παρουσιάστηκε στο κεφάλαιο 3 έδειξε ότι ένα αποτελεσματικό περιβάλλον οπτικοποίησης αλγορίθμων θα πρέπει να έχει τα εξής χαρακτηριστικά:

1. Να προωθεί την ενεργό εμπλοκή του μαθητή, τη διερεύνηση και τον πειραματισμό με την εκτέλεση του αλγορίθμου.
2. Να επιτρέπει την εκτέλεση του αλγορίθμου με δεδομένα που θα επιλέγει κάθε φορά ο χρήστης, ώστε να μπορεί να διερευνήσει όλες τις περιπτώσεις του αλγορίθμου (γενικές και ειδικές).
3. Κάθε οπτικοποίηση να εστιάζει και να αναδεικνύει τα ιδιαίτερα χαρακτηριστικά κάθε αλγορίθμου, τα οποία αποτελούν πηγή δυσκολιών ή παρανοήσεων για τους μαθητές.
4. Να φαίνεται ο πηγαίος κώδικας του αλγορίθμου, ώστε να μπορεί να συσχετιστεί με την οπτικοποίησή του.
5. Να επιτρέπει την τροποποίηση του κώδικα του αλγορίθμου και τον πειραματισμό με τη νέα οπτικοποίηση.
6. Το λογισμικό να εκτελείται μέσα από τον ιστό (web-based), ώστε να είναι ανεξάρτητο πλατφόρμας και προσβάσιμο μέσω φυλλομετρητή από κάθε ψηφιακή συσκευή (υπολογιστή, ταμπλέτα, ακόμη και έξυπνο κινητό).

Μέχρι σήμερα έχουν αναπτυχθεί πολλά συστήματα οπτικοποίησης αλγορίθμων μερικά από τα οποία εκτελούνται ακόμα και σε κινητά τηλέφωνα. Ωστόσο δεν υπάρχει κανένα σύστημα που να πληροί όλες τις παραπάνω προδιαγραφές διεθνώς. Ειδικά στην Ελλάδα δεν υπάρχει κάποιο άλλο σύστημα οπτικοποίησης αλγορίθμων διατυπωμένων στην γλώσσα προγραμματισμού που διδάσκεται στο Λύκειο. Το σύστημα υποστηρίζει

όλους τους αλγορίθμους που διδάσκονται στο γενικό και στο επαγγελματικό λύκειο διατυπωμένους στην ψευδογλώσσα που γνωρίζουν οι μαθητές.

## *7.3 Χαρακτηριστικά σχετικά με την έννοια του πίνακα*

Από την ανάλυση των αποτελεσμάτων των ερευνών που έγιναν για τις δυσκολίες που αντιμετωπίζουν οι μαθητές και οι φοιτητές στους πίνακες και στους αλγόριθμους ταξινόμησης, προέκυψαν τα εξής χαρακτηριστικά που πρέπει να έχει το DAVE:

1. Τα στοιχεία του πίνακα να μην είναι ενωμένα, αλλά να φαίνονται ως διακριτές οντότητες/μεταβλητές.
2. Να απεικονίζονται γραφικά οι δείκτες/μετρητές των επαναλήψεων ώστε να φαίνεται ο ρόλος τους κατά την εκτέλεση του αλγορίθμου. Με αυτόν τον τρόπο θα γίνει ορατή η αυτόματη αύξηση του μετρητή της επανάληψης Για…από…μέχρι (for), η οποία αποτελεί πηγή πολλών προβλημάτων για τους μαθητές (Danielsiek, 2012· Yamashita et al., 2016).
3. Να τονίζονται οι θέσεις των στοιχείων που εμπλέκονται σε κρίσιμα σημεία του αλγορίθμου (π.χ. στους αλγόριθμους αναζήτησης).
4. Να συσχετίζεται η τρέχουσα εντολή με την οπτικοποίηση της εκτέλεσης του αλγορίθμου.
5. Να μπορεί ο χρήστης να ελέγχει την ταχύτητα της οπτικοποίησης.
6. Να παρέχεται η δυνατότητα παύσης ή εκτέλεσης βήμα-βήμα του αλγορίθμου.
7. Να μπορεί ο χρήστης να εισάγει δεδομένα της επιλογής του ώστε να πειραματιστεί με ειδικές περιπτώσεις του αλγορίθμου.

## *7.4 Δυναμικά χαρακτηριστικά οπτικοποίησης*

Το DAVE υιοθετεί μια παιδαγωγική φιλοσοφία, η οποία θεωρεί τους μαθητές ενεργά υποκείμενα της μάθησης με στόχο την οικοδόμηση αποτελεσματικών αναπαραστάσεων για τις αλγοριθμικές δομές επίλυσης προβλημάτων με πίνακες και τα αντίστοιχα προγραμματιστικά αντικείμενα. Η εμπλοκή των μαθητών με στόχο την κατανόηση ενός αλγορίθμου βασίζεται στην αξιοποίηση του λάθους κατά τη διαδικασία της ανάπτυξης του αλγορίθμου από τους ίδιους τους μαθητές, και στην ανάδειξη των διαφορών ανάμεσα στα αναμενόμενα και στα παρατηρούμενα αποτελέσματα στην οθόνη.

Το σύστημα είναι σχεδιασμένο ώστε να υποστηρίζει τον πειραματισμό των μαθητών και τη διερευνητική μάθηση με στόχο την οικοδόμηση αλγορίθμων, μέσα από διαδικασίες δοκιμής και άμεσης παρατήρησης τους αποτελέσματος στην οθόνη του υπολογιστή. Ο παιδαγωγικός σχεδιασμός και η πρόταση εκπαιδευτικής αξιοποίησης του συστήματος DAVE υποστηρίζουν και εστιάζουν

- στην ανίχνευση των λογικών ή άλλων σφαλμάτων του μαθητή κατά την ανάπτυξη ή τροποποίηση ενός αλγορίθμου.
- στην ανάδειξη παρανοήσεων που σχετίζονται με προγραμματιστικές δομές και αντικείμενα.
- στην ανάδειξη των αναπαραστάσεων του μαθητή για τον αλγόριθμο ή τα κύρια μέρη αυτού.
- στην κατανόηση της δυναμικής συμπεριφοράς κάθε αλγορίθμου.

Ο μαθητής μπορεί να εισάγει τα δικά του δεδομένα , να ελέγξει την ταχύτητα της οπτικοποίησης, να σταματήσει την εκτέλεση του αλγορίθμου σε ένα συγκεκριμένο – κρίσιμο σημείο ή να εκτελέσει τον αλγόριθμο ανάστροφα (βήμα πίσω). Επιπλέον, το σύστημα παρέχει τη δυνατότητα στον μαθητή να τροποποιήσει την κωδικοποίηση του αλγορίθμου, να πειραματιστεί με την οπτικοποίηση της εκτέλεσης του δικού του αλγορίθμου και να εντοπίσει διαφορές και πιθανά λάθη.

Ο σχεδιασμός του λογισμικού δυναμικής οπτικοποίησης αλγορίθμων DAVE έγινε με βάση τα παραπάνω χαρακτηριστικά. Βασικός στόχος είναι η εμπλοκή και ο πειραματισμός του μαθητή με γνωστούς αλγορίθμους ταξινόμησης και αναζήτησης μέσα από την οπτικοποίησή τους. Το λογισμικό DAVE παρέχει τη δυνατότητα στο μαθητή να συσχετίσει κάθε εντολή του κώδικα, με κατάλληλες γραφικές αναπαραστάσεις δεδομένων έτσι ώστε να αναδεικνύονται τα ειδικά χαρακτηριστικά κάθε αλγορίθμου. Με βάση την προσέγγιση αυτή, ο μαθητής βρίσκεται σε άμεση και συνεχή επαφή με τον κώδικα του αλγορίθμου, καθώς κάθε αλλαγή στον κώδικα έχει άμεσο αντίκτυπο στην οπτικοποίηση. Ένα σημαντικό πλεονέκτημα του λογισμικού, που δεν συναντάται σε παρόμοια περιβάλλοντα της βιβλιογραφίας που μελετήθηκε (Βραχνός & Τζιμογιάννης, 2009· Sorva, Karavirta & Malmi, 2013· Urquiza–Fuentes & Velazquez–Iturbide, 2009), αποτελεί η δυνατότητα τροποποίησης του κώδικα του αλγορίθμου με ταυτόχρονη ενημέρωση της αντίστοιχης οπτικοποίησης.

Σύμφωνα με τον Stasko (1997), η υλοποίηση της οπτικοποίησης του αλγορίθμου από τον ίδιο το μαθητή είναι ο καλύτερος τρόπος να κατανοήσει τα βασικά

χαρακτηριστικά του υπό μελέτη αλγορίθμου. Ο μαθητής έχει τη δυνατότητα να παρακολουθεί την οπτικοποίηση της εκτέλεσης της δικής του έκδοσης του αλγορίθμου, κάτι που σημαίνει ότι οποιοδήποτε λάθος στον κώδικα θα φανεί άμεσα. Η ανατροφοδότηση από το σύστημα προσφέρει στο μαθητή μια συνολική εικόνα των ενεργειών του με στόχο να βοηθηθεί στην ανάπτυξη και βελτίωση του υπό μελέτη αλγορίθμου.

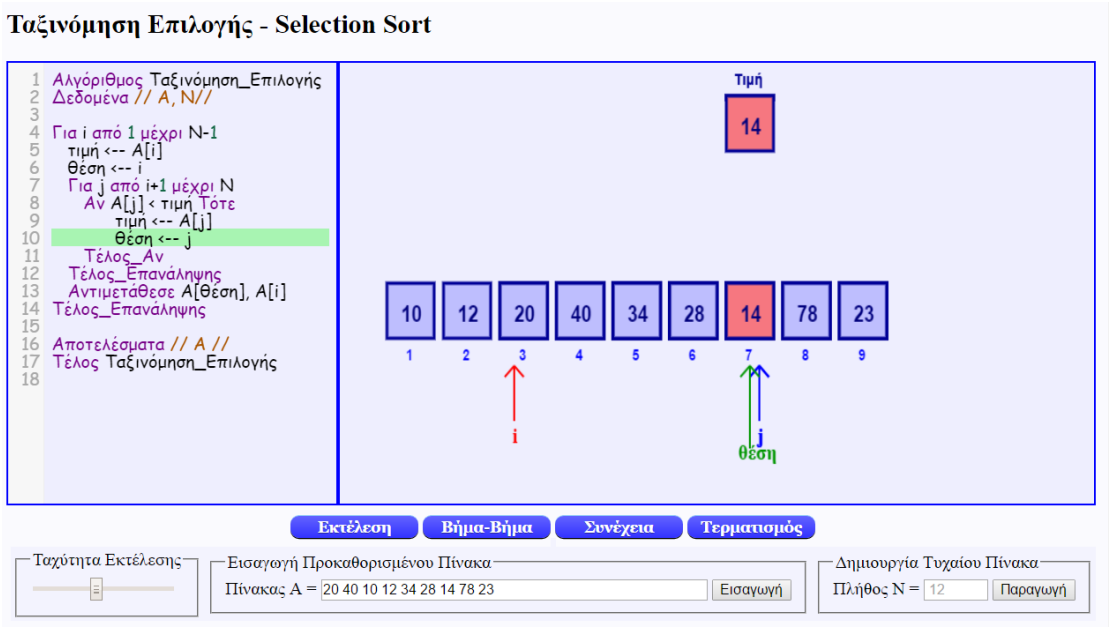
Η οπτικοποίηση ενός αλγορίθμου μέσω του DAVE δεν περιορίζεται στην απλή εμφάνιση των τιμών των μεταβλητών κατά την εκτέλεση, αλλά περιλαμβάνει την δυναμική οπτικοποίηση των δομών που προσομοιώνουν κατάλληλα τη λειτουργία του αλγορίθμου. Η επιλογή των μέσων αναπαράστασης έγινε έτσι ώστε ο μαθητής να μπορεί εύκολα να παρατηρήσει τις διαφορές μεταξύ των αναμενόμενων και των δικών του αποτελεσμάτων. Οι χρησιμοποιούμενες αναπαραστάσεις είναι διαφορετικές για κάθε αλγόριθμο. Για παράδειγμα η οπτικοποίηση της ταξινόμησης είναι διαφορετική από αυτή της αναζήτησης διότι οι προγραμματιστικές έννοιες και οι αλγοριθμικές λειτουργίες που πρέπει να τονιστούν σε κάθε περίπτωση δεν είναι οι ίδιες.

### *7.4.1 Σχεδιαστική επιλογή Ι : Μετακίνηση ή αντιγραφή;*

Μια από τις σχεδιαστικές επιλογές που έπρεπε να γίνουν για την οπτικοποίηση των αλγορίθμων ταξινόμησης ήταν η οπτικοποίηση της εντολής εκχώρησης και της αντιμετάθεσης των περιεχομένων δύο μεταβλητών. Στην περίπτωση της εκχώρησης θα πρέπει να τονίσουμε ότι δεν πρόκειται για μετακίνηση, παρανόηση που δημιουργείται σε πολλούς αρχάριους προγραμματιστές αλλά για αντιγραφή του περιεχομένου μιας μεταβλητής σε μια άλλη. Όπως φαίνεται παρακάτω στον αλγόριθμο ταξινόμησης επιλογής η εντολή εκχώρησης

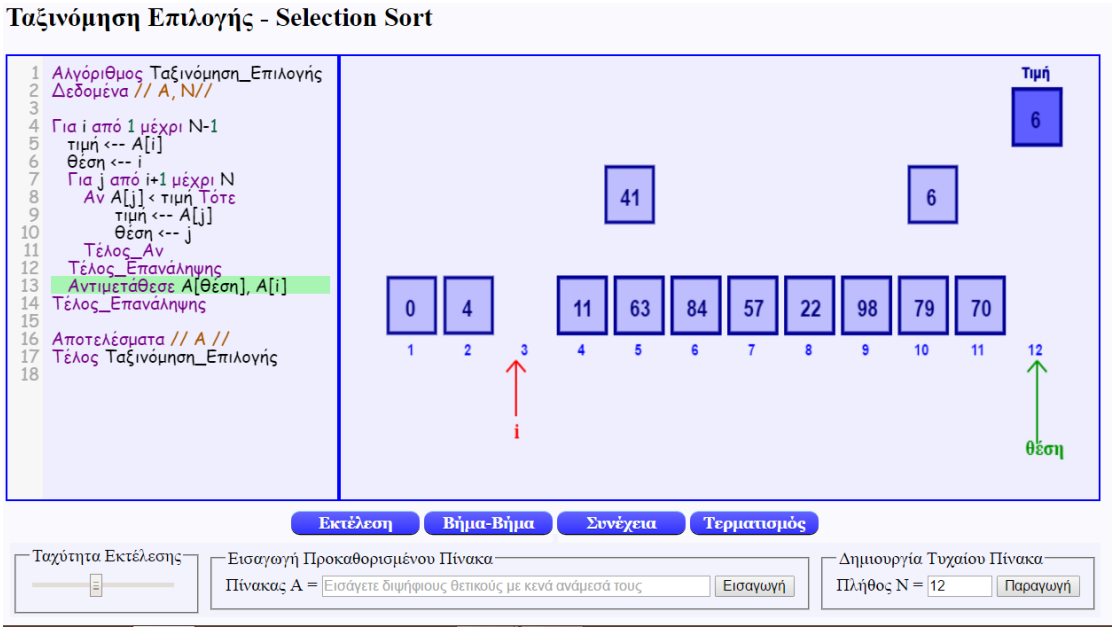
$$\textit{τιμή} \leftarrow A[\,j\,]$$

οπτικοποιείται με μετακίνηση ενός αντίγραφου του περιεχομένου της *A[j]* στη μεταβλητή *τιμή*.




**Σχήμα 7.2. Εκχώρηση τιμής ως αντιγραφή τιμής**

Αντίθετα η αντιμετάθεση των περιεχομένων δύο μεταβλητών δεν οπτικοποιείται με την ίδια λογική, διότι δεν μας ενδιαφέρει να οπτικοποιήσουμε το μοντέλο της νοητής μηχανής του υπολογιστή αλλά το φυσικό μοντέλο που προκύπτει από τις ενέργειες που θα έκανε κάποιος για να διατάξει μια σειρά αριθμών με τα χέρια.




**Σχήμα 7.3. Αμοιβαία αλλαγή των περιεχομένων των A[θέση] και A[i]**

Από παιδαγωγικής σκοπιάς, η επιλογή να οικοδομείται ένας αλγόριθμος ταξινόμησης πάνω σε δομικές ενέργειες που βασίζονται στην καθημερινή εμπειρία των μαθητών, έχει ως στόχο να χρησιμοποιηθούν νοητικά σχήματα και αναπαραστάσεις που είναι πιο κοντά στον δικό τους τρόπο σκέψης.

Για αυτό η αντιμετάθεση δύο στοιχείων οπτικοποιείται με την αμοιβαία αλλαγή θέσεων μεταξύ τους. Αξίζει να σημειωθεί ότι τη στιγμή που γίνεται η μετακίνηση οι αρχικές τους θέσεις μένουν κενές όπως φαίνεται στο Σχήμα 7.3, όπου παρουσιάζεται η οπτικοποίηση της εκτέλεσης του αλγόριθμου ταξινόμησης με επιλογή στο DAVE.

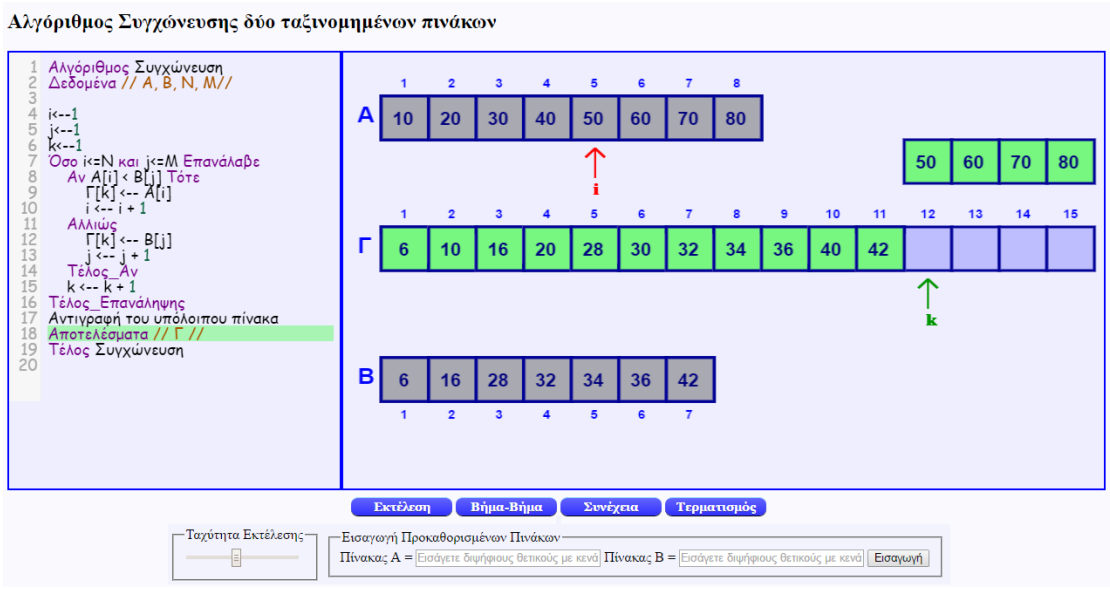
Το ίδιο ακριβώς θα συνέβαινε αν εκτελούσαμε μια αντιμετάθεση των περιεχομένων δύο γραμματοκιβωτίων με τα χέρια. Τα γράμματα δεν θα αντιγράφονταν αλλά θα μετακινούνταν και για ένα διάστημα τα δύο γραμματοκιβώτια θα έμεναν κενά.

### *7.4.2 Σχεδιαστική επιλογή ΙΙ : Το επίπεδο της αφαίρεσης*

Πέρα από την ομοιότητα με την αμοιβαία αλλαγή θέσεων δύο αντικειμένων στον πραγματικό κόσμο η απόκρυψη της υλοποίησης της αντιμετάθεσης δύο στοιχείων με χρήση προσωρινής μεταβλητής έχει και έναν άλλο σκοπό. Την οπτικοποίηση της αντιμετάθεσης σε ένα πιο υψηλό επίπεδο αφαίρεσης από αυτό των τριών εκχωρήσεων με χρήση βοηθητικής μεταβλητής. Στην εργασία των Corney, Lister & Teague (2011) η διάκριση ότι ένα τμήμα κώδικα εκτελεί αντιμετάθεση δύο μεταβλητών από τους μαθητές κατατάσσεται από τους συγγραφείς στο συνθετικό επίπεδο.

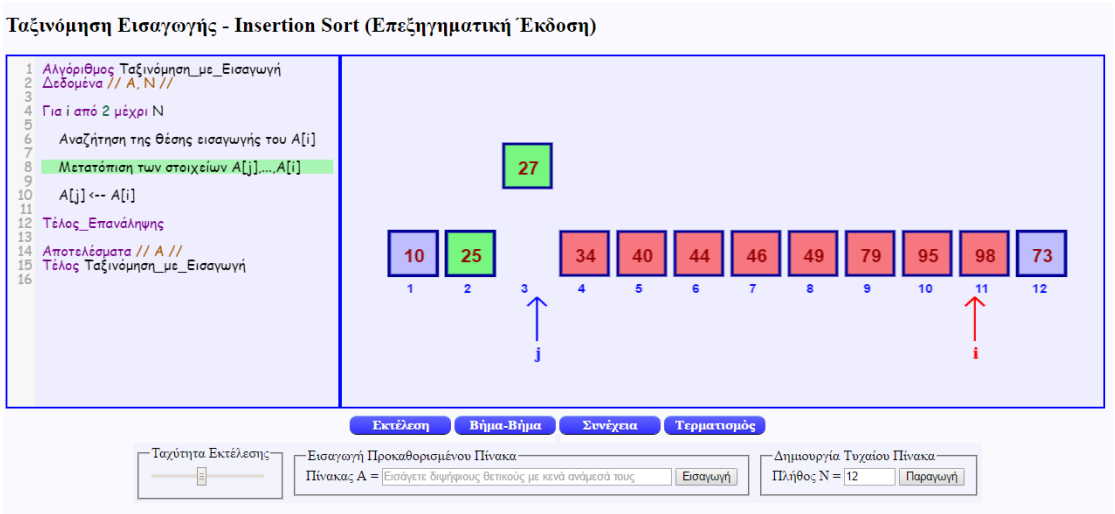
Το υψηλό επίπεδο αφαίρεσης ήταν μια από τις βασικές σχεδιαστικές αρχές του συστήματος όσον αφορά την κωδικοποίηση των αλγορίθμων και τα σημαντικά γεγονότα που κρίναμε ότι πρέπει να οπτικοποιηθούν. Στις επόμενες οθόνες δίνουμε δύο τέτοια παραδείγματα διατύπωσης αλγορίθμων σε ψευδογλώσσα υψηλού επιπέδου αφαίρεσης.

Στο Σχήμα 7.4 παρουσιάζεται η οπτικοποίηση του αλγορίθμου της συγχώνευσης, όπου το τελευταίο στάδιο της αντιγραφής του τμήματος του πίνακα που έχει "περισσέψει" παρουσιάζεται ως μια ενιαία εντολή τόσο στον κώδικα όσο και στην οπτικοποίηση. Μετά την εκτέλεση της εντολής *Αντιγραφή του υπόλοιπου πίνακα* αντιγράφεται το τμήμα 50 60 70 80 στο τέλος του πίνακα Γ. Πέρα από τα χρώματα σε αυτή την περίπτωση σημαντική βοήθεια για την κατανόηση της λειτουργίας του αλγόριθμου δίνουν οι δείκτες k, i.

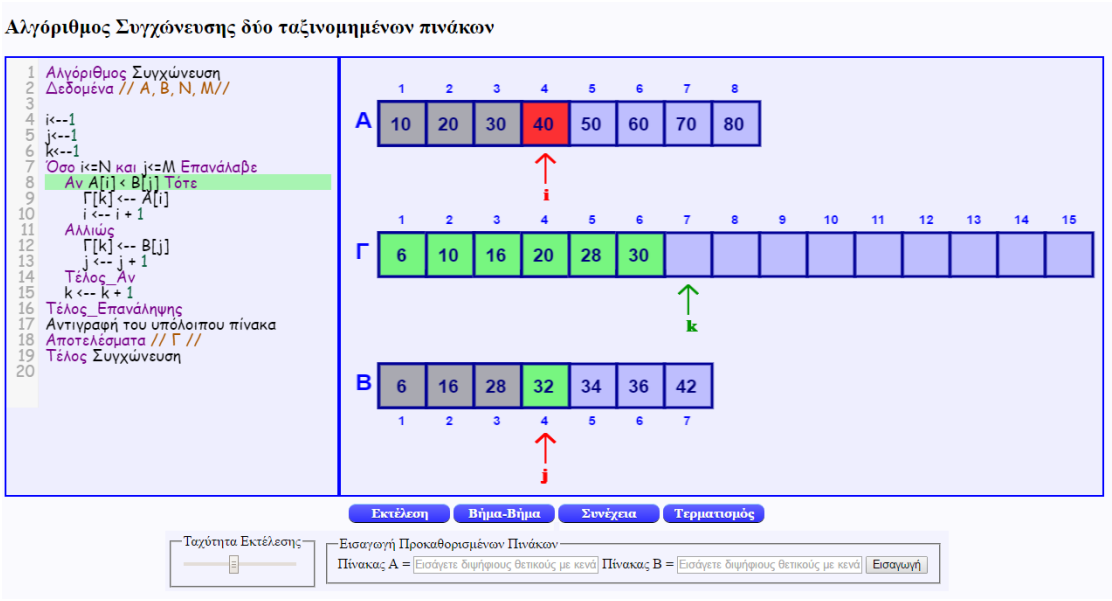



**Σχήμα 7.4. Τελικό στάδιο της συγχώνευσης δύο ταξινομημένων πινάκων Α και Β στον Γ**

Η δεύτερη περίπτωση διατύπωσης ενός αλγορίθμου σε υψηλό επίπεδο αφαίρεσης είναι αυτή του αλγορίθμου ταξινόμησης με εισαγωγή, που παρουσιάζεται στο Σχήμα 7.5. Ο αλγόριθμος διακρίνεται σε δύο μέρη, την αναζήτηση της θέσης στην οποία θα εισαχθεί το Α[i] και τη μετατόπιση δεξιά όλων των στοιχείων ώστε να δημιουργηθεί χώρος για την εισαγωγή του Α[i]. Η παραπάνω επεξηγηματική έκδοση του αλγορίθμου έχει σκοπό να δώσει τη βασική ιδέα της ταξινόμησης με εισαγωγή για αυτό το λόγο ο αλγόριθμος είναι διατυπωμένος σε υψηλότερο επίπεδο αφαίρεσης. Στη συνέχεια ο μαθητής μπορεί να πειραματιστεί με την γνωστή έκδοση του αλγορίθμου που είναι διατυπωμένη σε χαμηλότερο επίπεδο.




**Σχήμα 7.5. Αλγόριθμος ταξινόμησης με εισαγωγή σε υψηλό επίπεδο αφαίρεσης**




**Σχήμα 7.6. Το κεντρικό βήμα του αλγόριθμου συγχώνευσης στο DAVE**

## *7.5 Περιγραφή του συστήματος*

Το περιβάλλον δυναμικής οπτικοποίησης αλγορίθμων (Dynamic Algorithm Visualization Environment) DAVE φαίνεται στο Σχήμα 7.6, όπου παρουσιάζεται μια οπτικοποίηση του αλγορίθμου της συγχώνευσης μέσα από ένα πρόγραμμα πλοήγησης στον ιστό.

### *7.5.1 Διεπαφή και Λειτουργικότητα*

Όπως φαίνεται η διεπαφή του συστήματος είναι αρκετά απλή και αποτελείται από ένα και μόνο παράθυρο που χωρίζεται στις εξής περιοχές:

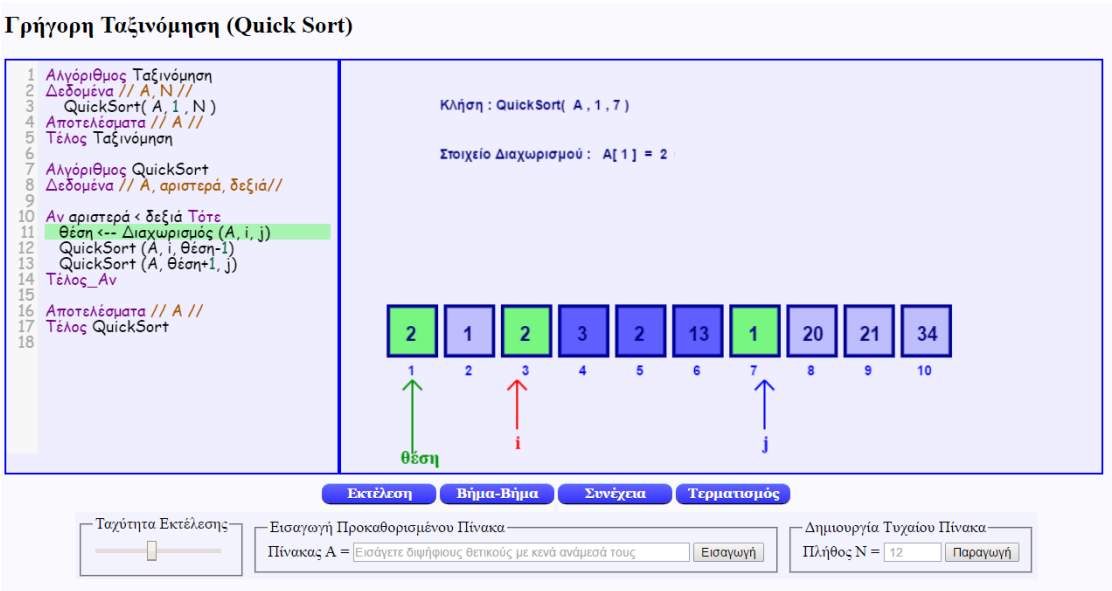
**Συντάκτης του πηγαίου κώδικα:** Είναι το αριστερό πλαίσιο της οθόνης, στο οποίο φαίνεται ο πηγαίος κώδικας του αλγόριθμου της οπτικοποίησης. Το εργαλείο υποστηρίζει μια μορφή ψευδογλώσσας στα ελληνικά η οποία έχει κάποιες ομοιότητες με την ψευδογλώσσα που χρησιμοποιείται στα μαθήματα προγραμματισμού του Λυκείου. Δίνεται η δυνατότητα στον χρήστη να τροποποιήσει τον πηγαίο κώδικα του αλγορίθμου σε συγκεκριμένα σημεία και να πειραματιστεί με την εκτέλεση της οπτικοποίησης. Ο χρήστης δεν μπορεί να κάνει σημαντικές αλλαγές στην φιλοσοφία του αλγορίθμου αλλά μπορεί να αλλάξει σημαντικές παραμέτρους όπως τα όρια και το βήμα των επαναλήψεων, τη συνθήκη σύγκρισης και τις εκφράσεις που ορίζουν τη θέση των στοιχείων που συγκρίνονται και αλλάζουν αμοιβαία θέση. Κατά την εκτέλεση του αλγορίθμου επισημαίνεται με χρώμα η εκτελούμενη εντολή.

**Περιοχή της οπτικοποίησης:** Είναι το μεγαλύτερο πλαίσιο δεξιά, όπου φαίνεται το αποτέλεσμα της οπτικοποίησης. Κάθε πίνακας αναπαρίσταται από μια ακολουθία από κελιά τα οποία δεν είναι ενωμένα μεταξύ τους. Ανάλογα με την εντολή που οπτικοποιείται δηλαδή αν είναι απλή απόδοση τιμής ή αμοιβαία αλλαγή μεταξύ μεταβλητών έχουμε αντιγραφή ή μετακίνηση κελιού.

**Πίνακας ελέγχου εκτέλεσης:** Ο πίνακας ελέγχου της εκτέλεσης του αλγορίθμου είναι το κάτω μέρος της οθόνης και υποστηρίζει τις ακόλουθες λειτουργίες:

- Εκτέλεση του αλγορίθμου μέχρι τέλος
- Εκτέλεση βήμα – βήμα του αλγορίθμου. Με κάθε πάτημα του πλήκτρου εκτελείται η επόμενη εντολή του αλγορίθμου.
- Παύση/Συνέχεια της εκτέλεσης
- Τερματισμός της εκτέλεσης
- Κυλιόμενη μπάρα για την αυξομείωση της ταχύτητας της εκτέλεσης του αλγορίθμου

**Είσοδος Δεδομένων:** Στο κάτω μέρος του παραθύρου ο χρήστης μπορεί να ορίσει τα δεδομένα με τα οποία θα εκτελεστεί ο αλγόριθμος. Στις περισσότερες περιπτώσεις ο χρήστης έχει δύο επιλογές όπως φαίνεται στον αλγόριθμο της γρήγορης ταξινόμησης του Σχήματος 7.7. Στο πεδίο *Εισαγωγή προκαθορισμένου πίνακα* ο χρήστης μπορεί να εισάγει τα στοιχεία του πίνακα με ένα κενό ανάμεσά τους, αν θέλει να μελετήσει τη συμπεριφορά του αλγορίθμου με διάφορα σύνολα δεδομένων.

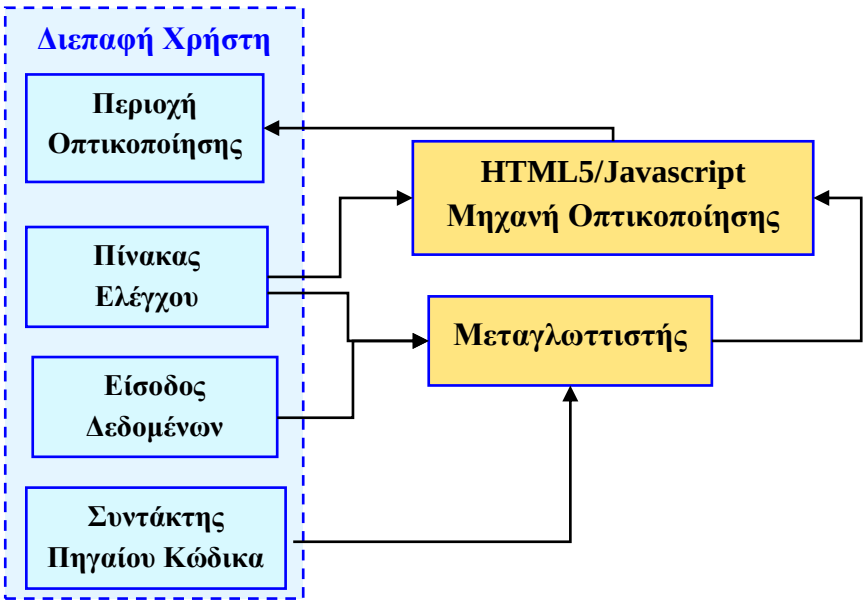



**Σχήμα 7.7. Ο αλγόριθμος της γρήγορης ταξινόμησης στο DAVE**

Ωστόσο υπάρχει και η δυνατότητα της δημιουργίας των στοιχείων ενός πίνακα με τυχαίο τρόπο, αρκεί να δοθεί το πλήθος τους, στο πεδίο *Δημιουργία τυχαίου πίνακα*.

### *7.5.2 Αρχιτεκτονική και υλοποίηση*

Το σύστημα οπτικοποίησης αλγορίθμων του DAVE αποτελείται ουσιαστικά από τρία τμήματα (Σχήμα 7.8): την διεπαφή με τον χρήστη, τον μεταγλωττιστή και την μηχανή οπτικοποίησης. Η γραφική διεπαφή είναι ουσιαστικά μια απλή ιστοσελίδα HTML5. Η μηχανή οπτικοποίησης είναι γραμμένη σε Javascript και για τον συντάκτη κώδικα έχουμε χρησιμοποιήσει τη βιβλιοθήκη CodeMirror (2018). Ο μεταγλωττιστής μετατρέπει τον πηγαίο κώδικα σε μια ενδιάμεση μορφή η οποία μπορεί να εκτελεστεί από την μηχανή οπτικοποίησης. Οι οπτικοποιήσεις που παράγονται είναι διαφορετικές για αλγορίθμους με διαφορετική λογική και εστιασμένες στα βασικά χαρακτηριστικά που αναδεικνύουν τη φιλοσοφία κάθε αλγορίθμου. Αυτά τα χαρακτηριστικά είναι που καθορίζουν τον χωρισμό των αλγορίθμων σε οικογένειες αλγορίθμων με κοινά χαρακτηριστικά οπτικοποίησης. Ανάλογα με την οικογένεια στην οποία ανήκει ένας αλγόριθμος (ταξινόμηση, αναζήτηση κ.α.) έχει έναν προκαθορισμένο σκελετό κώδικα τον οποίο ο χρήστης μπορεί να τροποποιήσει. Η μηχανή πρέπει να γνωρίζει εξαρχής ποιον αλγόριθμο θα οπτικοποιήσει ώστε να προετοιμάσει την κατάλληλη οπτικοποίηση.




**Σχήμα 7.8. Αρχιτεκτονική του συστήματος DAVE**

Δεν υπάρχει κάποιο γενικό πλαίσιο οπτικοποίησης για όλους τους αλγορίθμους, αλλά κάθε οπτικοποίηση είναι προσαρμοσμένη στα ιδιαίτερα χαρακτηριστικά κάθε αλγορίθμου. Για παράδειγμα η οπτικοποίησης της συγχώνευσης με αυτή της δυαδικής αναζήτησης είναι εντελώς διαφορετικές.

Τα περισσότερα συστήματα οπτικοποίησης είναι υλοποιημένα στη γλώσσα Java. Αυτό όμως σημαίνει ότι για να εκτελεστούν θα πρέπει να είναι εγκατεστημένη η σωστή έκδοση του περιβάλλοντος εκτέλεσης της Java (JRE). Ένα σημαντικό πρόβλημα είναι ότι τα περισσότερα προγράμματα πλοήγησης στον ιστό δεν υποστηρίζουν πλέον την εκτέλεση εφαρμογών Java applets. Αυτό έχει ως αποτέλεσμα την ραγδαία ανάπτυξη τεχνολογιών HTML5/Javascript για την ανάπτυξη εφαρμογών στον ιστό. Οι εφαρμογές αυτές έχουν τα παρακάτω σημαντικά πλεονεκτήματα:

- Είναι γρήγορες και αρκετά ελαφριές. Ακόμα και αν κολλήσουν ή δεν ανταποκρίνονται αρκεί να κάνουμε ανανέωση στην ιστοσελίδα και όλα ξεκινούν από την αρχή.
- Δεν χρειάζεται να εγκαταστήσουμε ή να κατεβάσουμε απολύτως τίποτα. Οι εφαρμογές αυτές έχουν ενσωματωμένη υποστήριξη σε όλα τα προγράμματα πλοήγησης του ιστού.
- Οι εφαρμογές αυτές εκτελούνται σε οποιαδήποτε πλατφόρμα μπορεί να εκτελεστεί ένα πρόγραμμα πλοήγησης τους ιστού (browser), δηλαδή παντού (υπολογιστές, ταμπλέτες, κινητά τηλέφωνα).
- Η εφαρμογή είναι αναρτημένη στον ιστό και άρα προσβάσιμη από οπουδήποτε και οποιονδήποτε.
- Η εφαρμογή εκτελείται πλήρως στον υπολογιστή του εξυπηρετούμενου (client-based) και δεν έχει καμία εξάρτηση από κάποιον απομακρυσμένο διακομιστή.

## *7.6 Περιεχόμενο*

Το λογισμικό υποστηρίζει την οπτικοποίηση των αλγορίθμων ταξινόμησης και αναζήτησης που αναφέρονται στα σχολικά βιβλία της γενικής και επαγγελματικής εκπαίδευσης.

### *7.6.1 Αλγόριθμος ταξινόμησης ευθείας ανταλλαγής*

Ο αλγόριθμος ταξινόμησης ευθείας ανταλλαγής (straight exchange sort) βασίζεται στη σύγκριση και την αντιμετάθεση γειτονικών στοιχείων ενός πίνακα σε περίπτωση που αυτά είναι σε λάθος σειρά. Η ταξινόμηση συνεχίζεται μέχρι τα στοιχεία του πίνακα να διαταχθούν στην επιθυμητή σειρά (αύξουσα/φθίνουσα). Η γενική ιδέα είναι ότι σε κάθε πέρασμα του αλγορίθμου αναδύεται το αμέσως μικρότερο στοιχείο στην επιφάνεια, για αυτό ο αλγόριθμος είναι γνωστός και ως αλγόριθμος ταξινόμησης φυσαλίδας (bubble sort).

Σε πολλές περιπτώσεις, ο αλγόριθμος επιλέγεται για την μύηση των αρχάριων προγραμματιστών στους αλγόριθμους ταξινόμησης, κυρίως, λόγω της σύντομης κωδικοποίησής του. Όπως φαίνεται στον Αλγόριθμο 7.1, ο αλγόριθμος μπορεί να διατυπωθεί με μόλις τρεις εντολές κώδικα, κάτι που διευκολύνει τη διαχείριση και τον γρήγορο προγραμματισμό του.

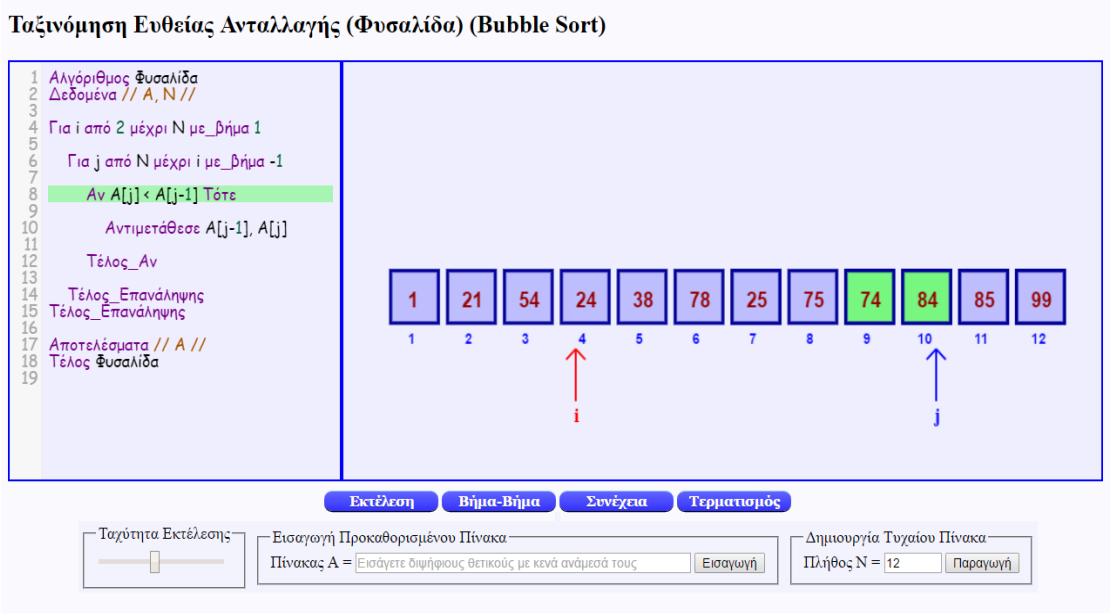
```
Αλγόριθμος ΤαξινόμησηΕυθείαςΑνταλλαγής
Δεδομένα // A, N //
    Για i από 2 μέχρι N
        Για j από N μέχρι i με βήμα -1
            Αν A[ j ] < A[ j-1 ] Τότε
                Αντιμετάθεσε A[ j ], A[ j-1 ]
            Τέλος_Αν
        Τέλος_Επανάληψης
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος ΤαξινόμησηΕυθείαςΑνταλλαγής
```

**Αλγόριθμος 7.1. Ταξινόμηση Ευθείας Ανταλλαγής (Bubble Sort)**

Ωστόσο όπως φάνηκε και από την έρευνα που εκπονήθηκε στο πλαίσιο της διατριβής, οι μαθητές παρουσιάζουν σημαντικές δυσκολίες στην κατανόηση του αλγορίθμου αυτού. Παρόλα αυτά ο αλγόριθμος ταξινόμησης ευθείας ανταλλαγής έχει επικρατήσει ως ο βασικός αλγόριθμος ταξινόμησης στην δευτεροβάθμια εκπαίδευση.

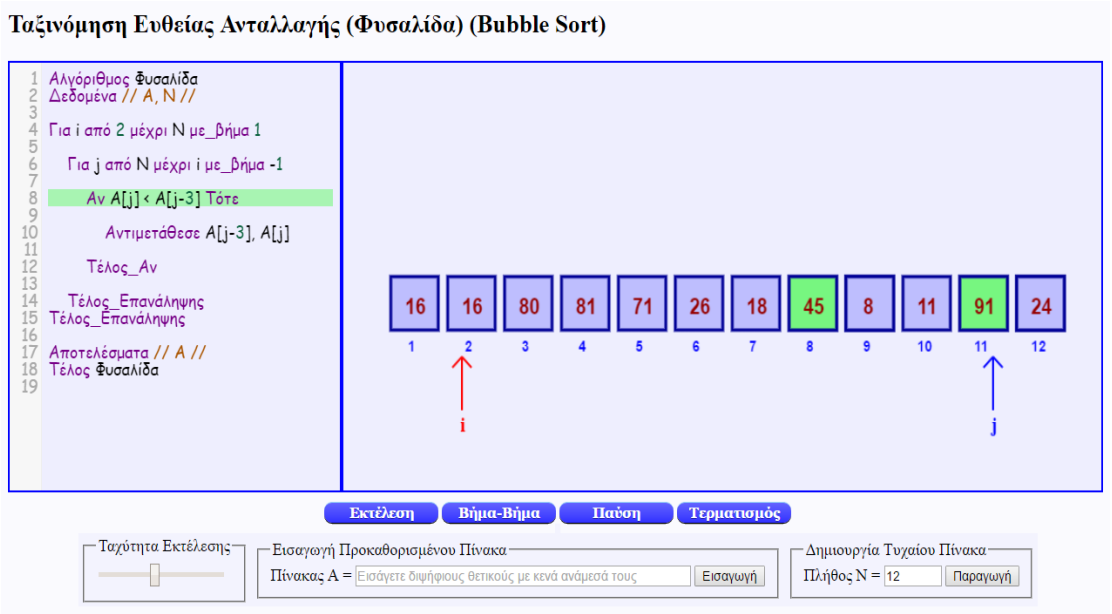
Ο πειραματισμός με την οπτικοποίηση του αλγορίθμου μπορεί να βοηθήσει τους μαθητές να διακρίνουν τα βασικά χαρακτηριστικά της λειτουργίας του και να τα συσχετίσουν με συγκεκριμένα σημεία του κώδικα. Για παράδειγμα, να διακρίνουν ότι

η εξωτερική επανάληψη με τον δείκτη i ορίζει το πλήθος των περασμάτων, δηλαδή το πλήθος των στοιχείων που θα ταξινομηθούν.




**Σχήμα 7.9. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής στο DAVE**

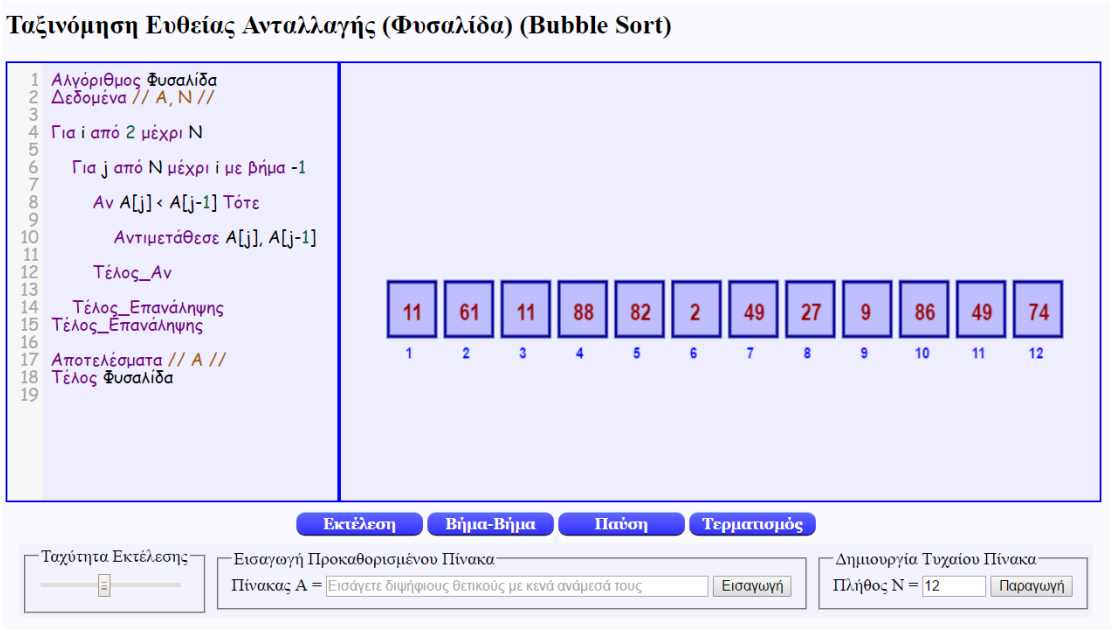
Στο σχήμα 7.9 φαίνεται ένα στιγμιότυπο της εκτέλεσης του αλγορίθμου όπου j=10 και i=4. Τα στοιχεία στις θέσεις 9 και 10 έχουν επισημανθεί με πράσινο χρώμα που σημαίνει ότι βρίσκονται ήδη σε αύξουσα σειρά και δεν χρειάζεται να αλλάξουν θέση μεταξύ τους. Το λογισμικό παρέχει τη δυνατότητα να τροποποιηθεί ο κώδικας του αλγορίθμου.



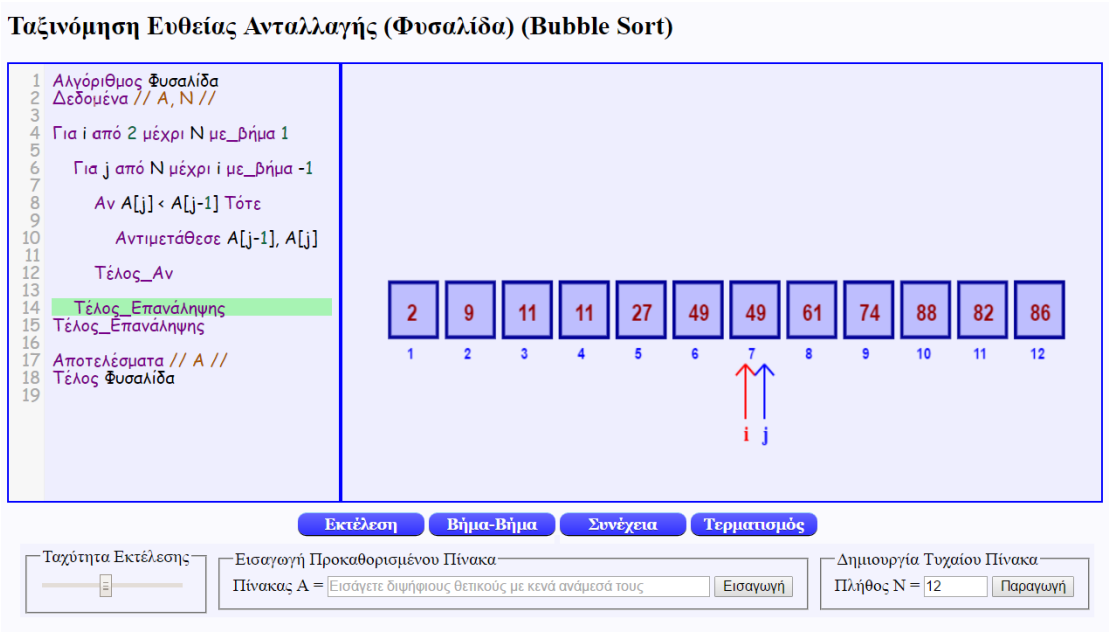

**Σχήμα 7.10. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής στο DAVE**

Για παράδειγμα στο Σχήμα 7.10 συγκρίνονται σε κάθε βήμα τα στοιχεία ανά τρεις θέσεις, δηλαδή το 12ο με το 9ο , το 11ο με το 8ο κ.ο.κ.

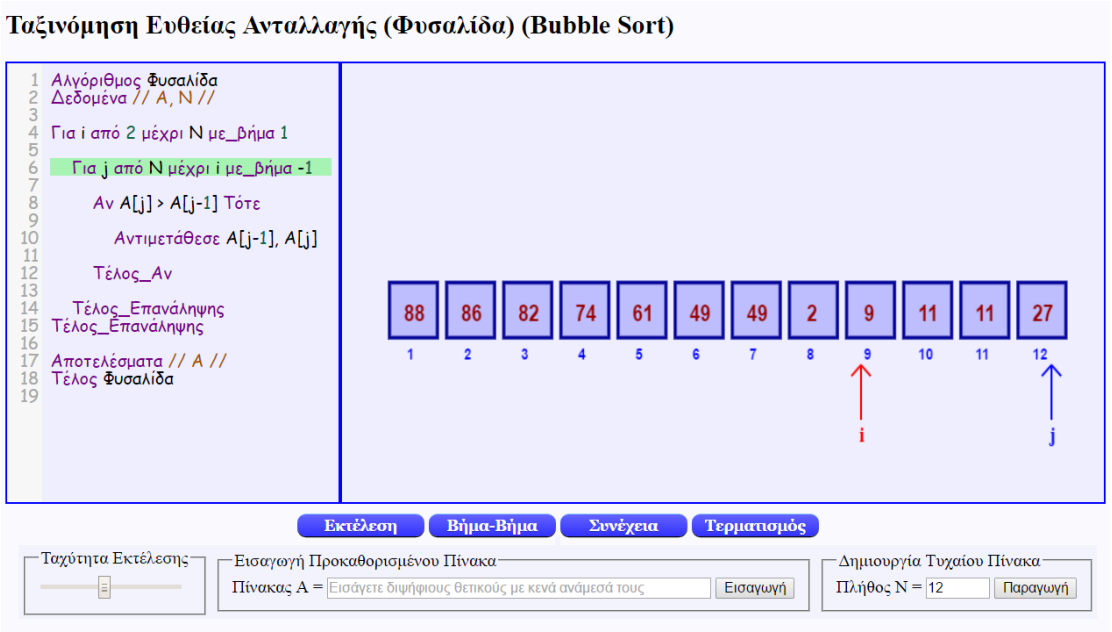
Μια σημαντική δυνατότητα που έχει το λογισμικό είναι ότι επιτρέπει την εκτέλεση του αλγορίθμου μέχρι ένα σημείο, την τροποποίηση του κώδικα και στην συνέχεια την επανεκτέλεση του νέου αλγορίθμου στα υπάρχοντα δεδομένα. Με αυτόν τον τρόπο μπορούμε να ταξινομήσουμε ένα τμήμα του πίνακα σε φθίνουσα σειρά και ένα άλλο σε αύξουσα, όπως φαίνεται στο Σχήμα 7.11. Αρχικά δημιουργείται ένας πίνακας για ταξινόμηση στον οποίο τα στοιχεία είναι διατεταγμένα με τυχαίο τρόπο. Εκτελούμε τον αλγόριθμο ευθείας ανταλλαγής μέχρι το σημείο που θέλουμε, για παράδειγμα μέχρι να ανέβουν και τα δύο 49 (Σχήμα 7.12).




**Σχήμα 7.11. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής στο DAVE**




**Σχήμα 7.12. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής στο DAVE**




**Σχήμα 7.13. Αλγόριθμος Ταξινόμησης Ευθείας Ανταλλαγής στο DAVE**

Στη συνέχεια αλλάζουμε τον κώδικα του αλγορίθμου έτσι ώστε να εκτελεί φθίνουσα και όχι αύξουσα ταξινόμηση και επιλέγουμε εκτέλεση από την αρχή. Αυτό σημαίνει ότι οι δείκτες i,j θα αρχικοποιηθούν αλλά ο πίνακας θα συνεχίσει από εκεί που έμεινε, δηλαδή με τα πρώτα 7 στοιχεία του σε αύξουσα σειρά.

Στη συνέχεια σταματάμε την εκτέλεση του αλγορίθμου στο σημείο που θέλουμε. Αυτό θα ήταν πιο εύκολο αν χρησιμοποιούσαμε την επιλογή *βήμα-βήμα* ώστε να ελέγχουμε καλύτερα την ταχύτητα της εκτέλεσης. Θα μπορούσαμε να πετύχουμε το

ίδιο με παρέμβαση μόνο στον κώδικα ορίζοντας το άνω όριο της εξωτερικής επανάληψης ίσο με 8. Το αποτέλεσμα είναι το ίδιο, δηλαδή το πρώτο τμήμα του πίνακα ταξινομημένο σε φθίνουσα σειρά και τα τελευταία στοιχεία στις ίδιες σχετικές θέσεις μεταξύ τους, δηλαδή σε αύξουσα σειρά.

### *7.6.2 Αλγόριθμος ταξινόμησης με επιλογή*

Ο αλγόριθμος ταξινόμησης με επιλογή (selection sort) είναι μια επαναλαμβανόμενη εύρεση του ελάχιστου στοιχείου του πίνακα, το οποίο τοποθετείται κάθε φορά στη σωστή θέση. Μοιάζει με τον αλγόριθμο ευθείας ανταλλαγής γιατί τα μικρότερα στοιχεία συγκεντρώνονται στην αρχή του πίνακα κατά την εκτέλεση του αλγορίθμου. Η διαφορά είναι ότι σε κάθε πέρασμα έχουμε μια μόνο αντιμετάθεση, του ελάχιστου στοιχείου και του στοιχείου που βρίσκεται στη θέση στην οποία θα τοποθετηθεί το ελάχιστο.

```
Αλγόριθμος ΤαξινόμησηΕπιλογής
Δεδομένα // A, N //
    Για i από 1 μέχρι N–1
        τιμή ← A[i]
        θέση ← i
        Για j από i+1 μέχρι N
            Αν A[ j ] < τιμή Τότε
                τιμή ← A[i]
                θέση ← i
            Τέλος_Αν
        Τέλος_Επανάληψης
        Αντιμετάθεσε A[ θέση ], A[ i ]
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος ΤαξινόμησηΕπιλογής
```

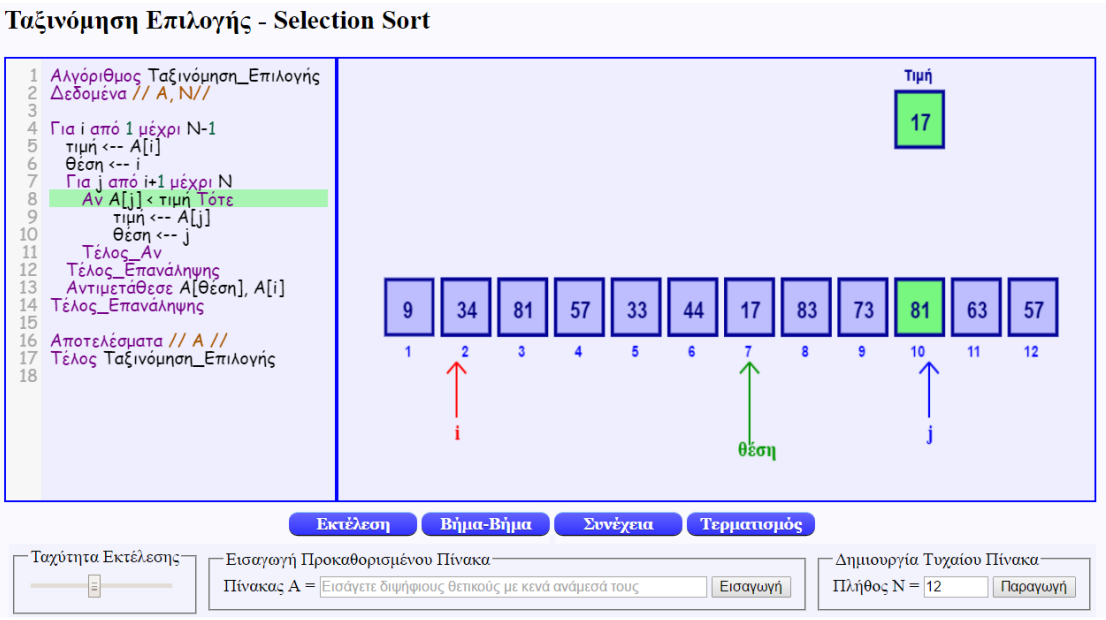
**Αλγόριθμος 7.2. Ταξινόμηση με επιλογή (Selection Sort)**

Ο αλγόριθμος ταξινόμησης με επιλογή θεωρείται από τους πιο απλούς αλγορίθμους γιατί μπορεί να οικοδομηθεί πάνω στον αλγόριθμο εύρεσης της ελάχιστης τιμής τον οποίο γνωρίζουν οι μαθητές. Αν για την εύρεση της ελάχιστης τιμής ορίσουμε ένα νέο αλγόριθμο τότε η κωδικοποίηση της ταξινόμησης με επιλογή μπορεί να γίνει πολύ πιο απλή και επεξηγηματική, όπως φαίνεται στον Αλγόριθμο 7.3.

```
Αλγόριθμος ΤαξινόμησηΕπιλογής
Δεδομένα // A, N //
    Για i από 1 μέχρι N–1
        θέση ← ΘέσηΕλαχίστου( A, i, N )
        Αντιμετάθεσε A[ θέση ], A[ i ]
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος ΤαξινόμησηΕπιλογής

Αλγόριθμος ΘέσηΕλαχίστου
Δεδομένα // Πίνακας, i, N //
    τιμή ← A[i]
    θέση ← i
    Για j από i+1 μέχρι N
        Αν A[ j ] < τιμή Τότε
            τιμή ← A[i]
            θέση ← i
        Τέλος_Αν
    Τέλος_Επανάληψης
Αποτελέσματα // θέση //
Τέλος ΘέσηΕλαχίστου
```

**Αλγόριθμος 7.3. Επεξηγηματική έκδοση της Ταξινόμησης με Επιλογή**



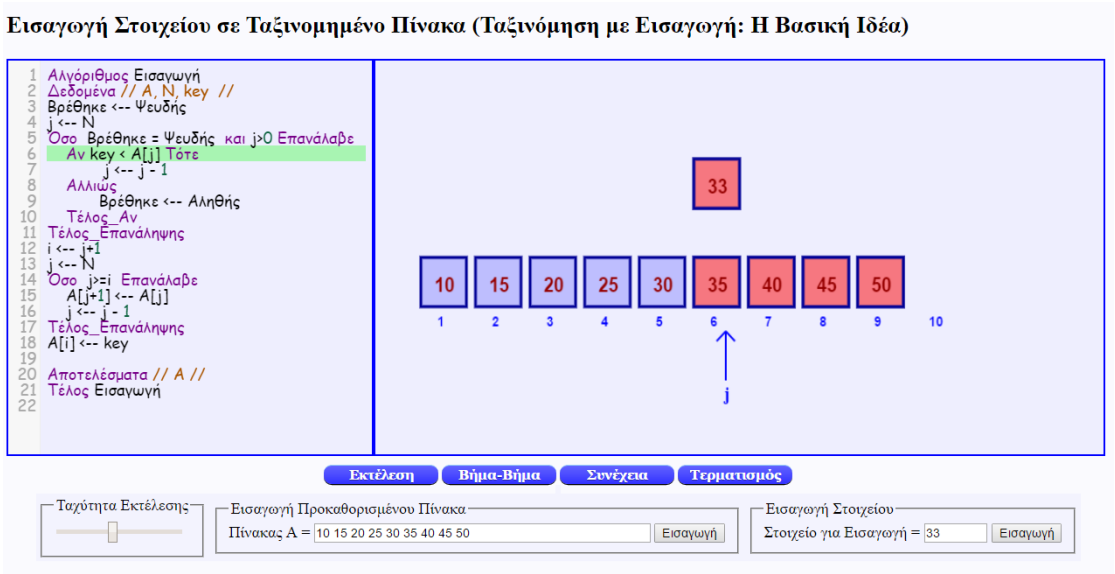

**Σχήμα 7.14. Αλγόριθμος Ταξινόμησης με επιλογή στο DAVE**

Ένα ακόμα σημαντικό πλεονέκτημα του αλγορίθμου πέρα από την απλότητά του είναι ότι προκύπτει άμεσα από τις εμπειρίες των μαθητών (Βραχνός & Τζιμογιάννης, 2018· 2016). Δηλαδή είναι από τους αλγορίθμους που χρησιμοποιούν οι άνθρωποι στην καθημερινότητα για να ταξινομήσουν μια σειρά από αντικείμενα, χωρίς να τον έχουν διδαχθεί. Στο Σχήμα 7.14 δίνεται μια οθόνη του λογισμικού από την εκτέλεση του αλγόριθμου με επιλογή για έναν πίνακα 12 στοιχείων.

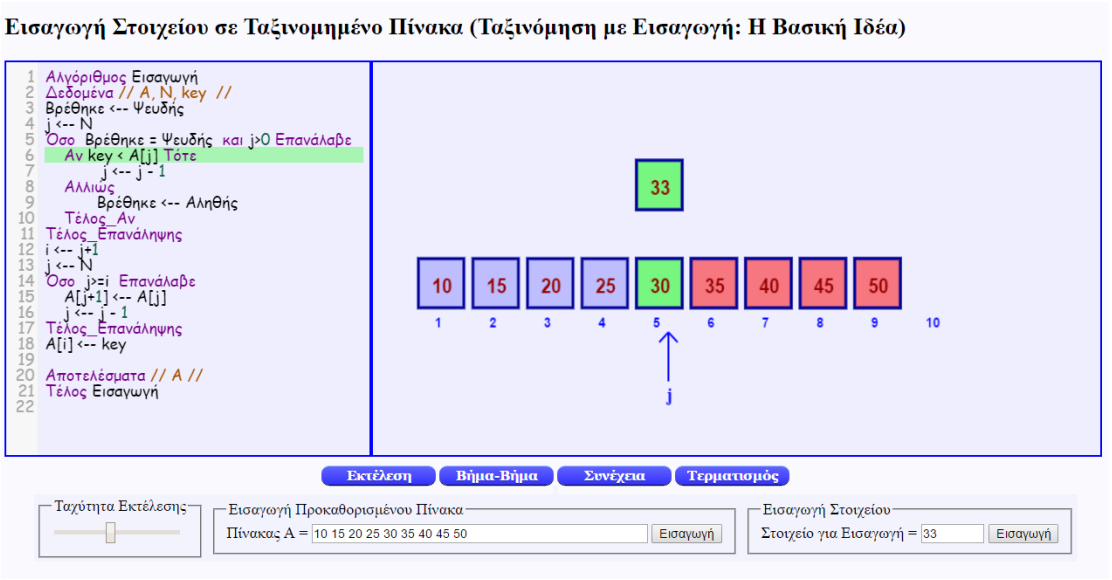
### *7.6.3 Αλγόριθμος ταξινόμησης με εισαγωγή*

Ο αλγόριθμος ταξινόμησης με εισαγωγή αποτελεί την αυθαίρετη επιλογή των μαθητών αν τους ζητηθεί να ταξινομήσου μια σειρά από αντικείμενα πάνω στα οποία ορίζεται μια σχέση διάταξης (Βραχνός & Τζιμογιάννης, 2018· 2016). Όσοι άνθρωποι ταξινομούν μια σειρά αντικείμενα με απτό τρόπο συνήθως χρησιμοποιούν αυτόν τον αλγόριθμο, κυρίως σε περιπτώσεις που δεν έχουν όλα τα αντικείμενα από την αρχή στη διάθεσή τους.

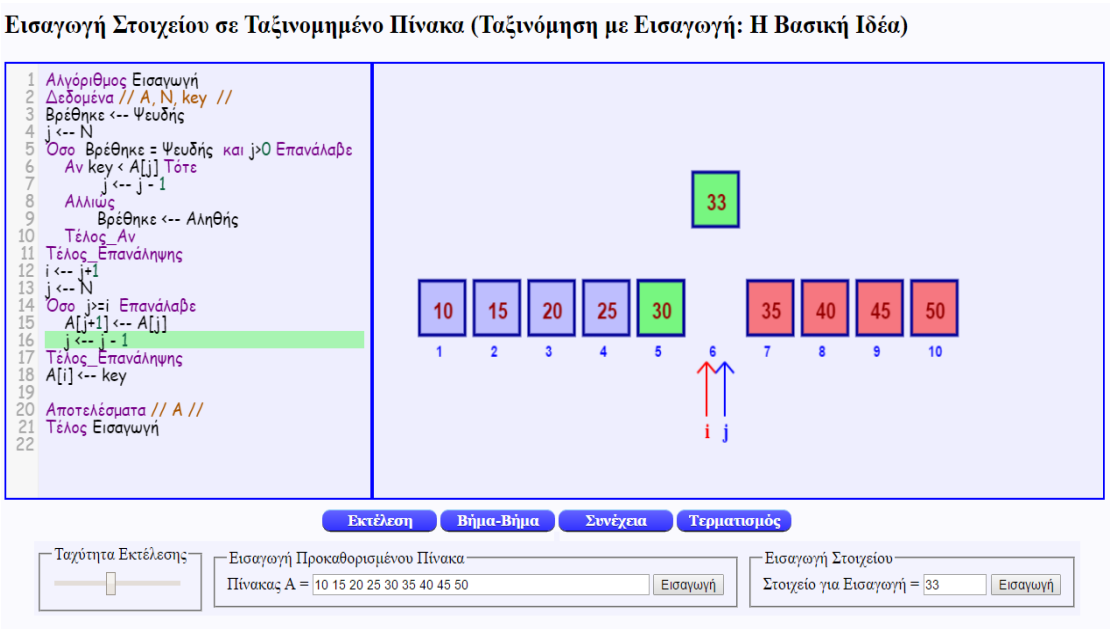
Για παράδειγμα όταν έχουμε συνεχή ροή δεδομένων και θέλουμε αυτά να είναι ανά πάσα στιγμή διατεταγμένα σε κάποια σειρά, πρέπει να εισάγουμε το νέο δεδομένο στην κατάλληλη θέση έτσι ώστε να μην χαλάσει η σειρά. Άρα ο αλγόριθμος της ταξινόμησης με εισαγωγή αποτελείται από δύο βασικά βήματα. Αρχικά γίνεται αναζήτηση της θέσης εισαγωγής του νέου στοιχείου (Σχήμα 7.16).




**Σχήμα 7.15. Αλγόριθμος Ταξινόμηση με Εισαγωγή: Αναζήτηση στο DAVE**




**Σχήμα 7.16. Ταξινόμηση με Εισαγωγή: Εύρεση θέσης εισαγωγής στο DAVE**




**Σχήμα 7.17. Αλγόριθμος Ταξινόμηση με Εισαγωγή: Μετακίνηση στοιχείων στο DAVE**

Στη συνέχεια αφού βρεθεί η θέση στην οποία πρέπει να εισαχθεί το νέο στοιχείο, μετακινούνται όλα τα στοιχεία μια θέση δεξιά ώστε να ανοίξει μια κενή θέση για το νέο στοιχείο (Σχήμα 7.17). Το μόνο που μένει τώρα είναι να εισαχθεί το νέο στοιχείο στη σωστή θέση.

Το σύστημα επιτρέπει τη διαδοχική εισαγωγή πολλών στοιχείων στον πίνακα, αν εκτελέσουμε τον αλγόριθμο περισσότερες από μια φορές. Σε κάθε εκτέλεση χρησιμοποιείται ο πίνακας που προέκυψε από την προηγούμενη εισαγωγή. Στο πεδίο *στοιχείο για εισαγωγή* ο μαθητής δίνει το νέο στοιχείο που θέλει να εισαχθεί στον

πίνακα. Με διαδοχικές εκτελέσεις του παραπάνω κώδικα ο μαθητής μπορεί να οικοδομήσει τη λειτουργία του αλγόριθμου ταξινόμησης με εισαγωγή πάνω στις δομικές λειτουργίες της εισαγωγής και της ολίσθησης τμήματος του πίνακα. Στη συνέχεια μπορεί να δοθεί μια περιγραφή του αλγορίθμου σε ένα πιο υψηλό επίπεδο αφαίρεσης.

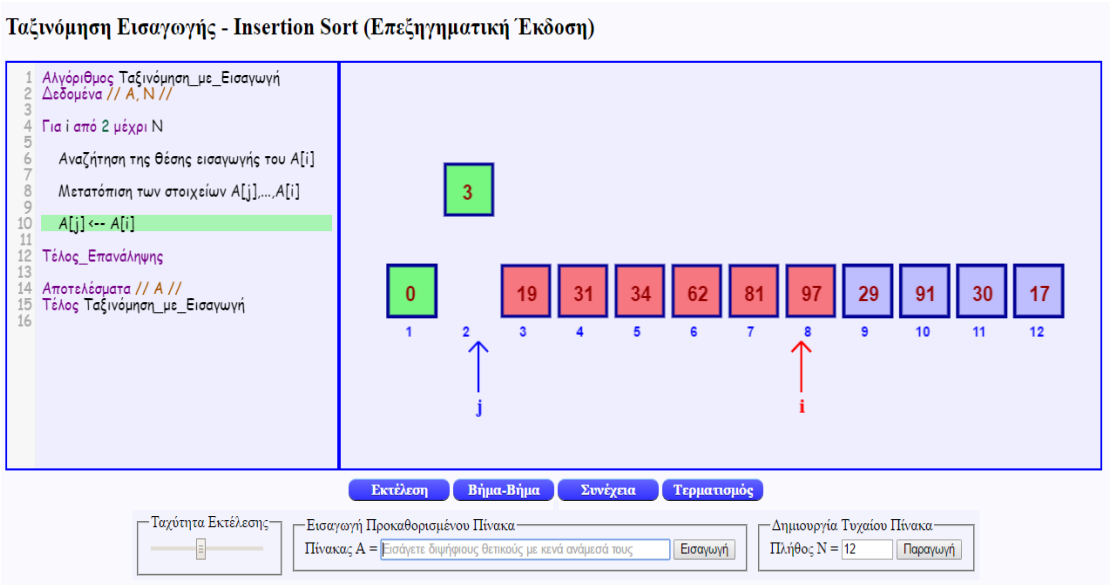
```
Αλγόριθμος ΤαξινόμησηΕισαγωγής
Δεδομένα // A, N //
    Για i από 2 μέχρι N
        θέση ← Αναζήτηση της θέσης εισαγωγής του A[ i ]
        Μετατόπιση των στοιχείων A[ j ], …, A[ i ]
        A[ j ] ← A[ i ]
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος ΤαξινόμησηΕισαγωγής
```

**Αλγόριθμος 7.4. Επεξηγηματική έκδοση της ταξινόμησης με εισαγωγή**

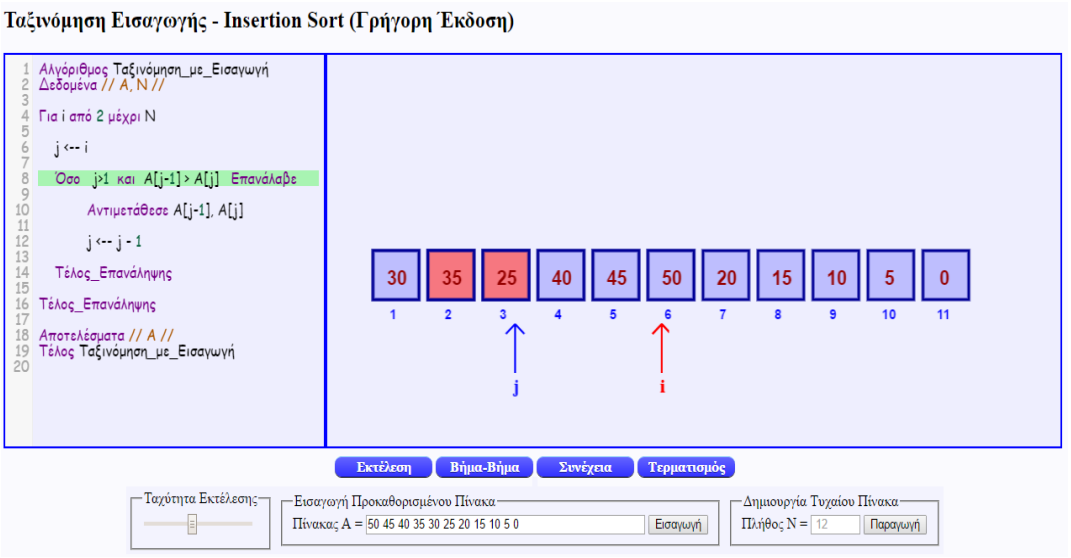
Στο τέλος μπορεί να δοθεί η τελική σύντομη έκδοση του αλγορίθμου στην οποία η μετατόπιση γίνεται ταυτόχρονα με την αναζήτηση.

```
Αλγόριθμος ΤαξινόμησηΕισαγωγής
Δεδομένα // A, N //
    Για i από 2 μέχρι N
        j ← i
        Όσο j > 1 και A[ j–1 ] > A[ j ] Επανάλαβε
            Αντιμετάθεσε A[ j–1 ], A[ j ]
            j ← j – 1
        Τέλος_Επανάληψης
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος ΤαξινόμησηΕισαγωγής
```

**Αλγόριθμος 7.5. Ταξινόμηση με εισαγωγή (Insertion Sort)**



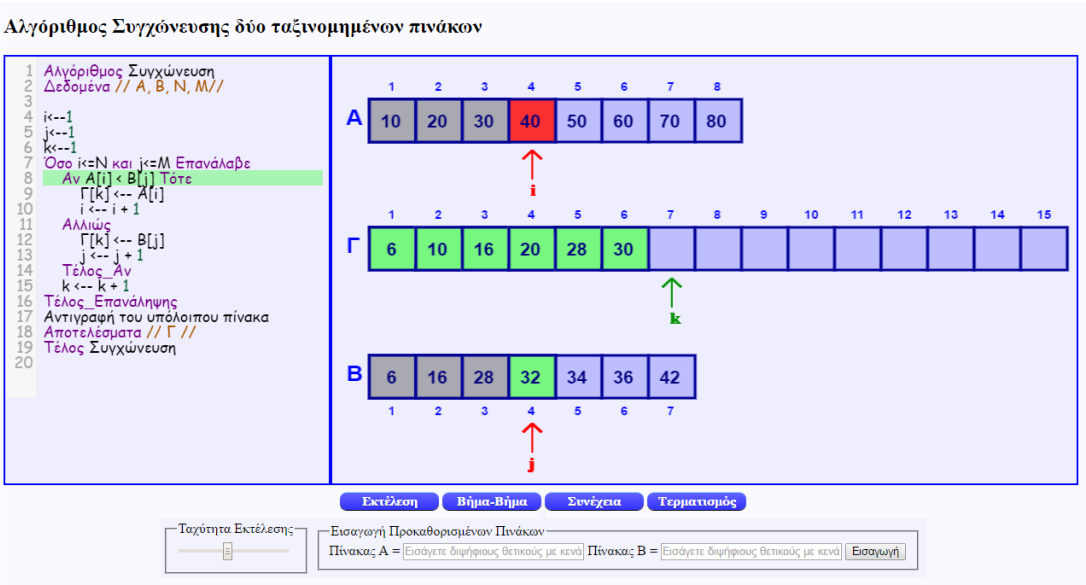

**Σχήμα 7.18. Ταξινόμηση με Εισαγωγή: Επεξηγηματική έκδοση στο DAVE**




**Σχήμα 7.19. Αλγόριθμος Ταξινόμησης με Εισαγωγή: Τελική έκδοση στο DAVE**

### *7.6.4 Αλγόριθμος συγχώνευσης*

Ο αλγόριθμος της συγχώνευσης δέχεται δύο ταξινομημένους πίνακες και τους ενώνει σε έναν νέο ταξινομημένο πίνακα, χωρίς να χρησιμοποιηθεί κάποιος αλγόριθμος ταξινόμησης. Ο αλγόριθμος της συγχώνευσης αξιοποιεί το γεγονός ότι οι αρχικοί πίνακες είναι ταξινομημένοι, έτσι ώστε να εκτελέσει την συγχώνευση με γραμμική πολυπλοκότητα O(M+N), όπου M,N τα μεγέθη των δύο πινάκων. Το βασικό βήμα του αλγορίθμου φαίνεται στο Σχήμα 7.20, όπου συγκρίνεται το αμέσως μικρότερο στοιχείο του πίνακα Α με το αντίστοιχο στοιχείο του Β ώστε το μικρότερο εκ των δυο (το 40) να τοποθετηθεί στην 7η θέση του πίνακα Γ.




**Σχήμα 7.20. Αλγόριθμος Συγχώνευσης δύο ταξινομημένων πινάκων στο DAVE**

**Αλγόριθμος** Συγχώνευση
**Δεδομένα** // A, B, N, M //
i ← 1
j ← 1
k ← 1
**Όσο** i ≤ N **και** j ≤ M **Επανάλαβε**
**Αν** A[ i ] < B[ j ] **Τότε**
Γ[ k ] ← A[ i ]
i ← i + 1
**Αλλιώς**
Γ[ k ] ← B[ j ]
j ← j + 1
**Τέλος_Αν**
k ← k + 1
**Τέλος_Επανάληψης**
*Αντιγραφή του υπόλοιπου πίνακα*
**Αποτελέσματα** // A //
**Τέλος** Συγχώνευση

**Αλγόριθμος 7.6. Συγχώνευση δύο ταξινομημένων πινάκων**

### *7.6.5 Αλγόριθμος διαχωρισμού*

Ο αλγόριθμος διαχωρισμού εξετάζεται ξεχωριστά, καθώς αποτελεί βασική λειτουργία του αλγορίθμου της γρήγορης ταξινόμησης. Ο αλγόριθμος διαχωρίζει τον

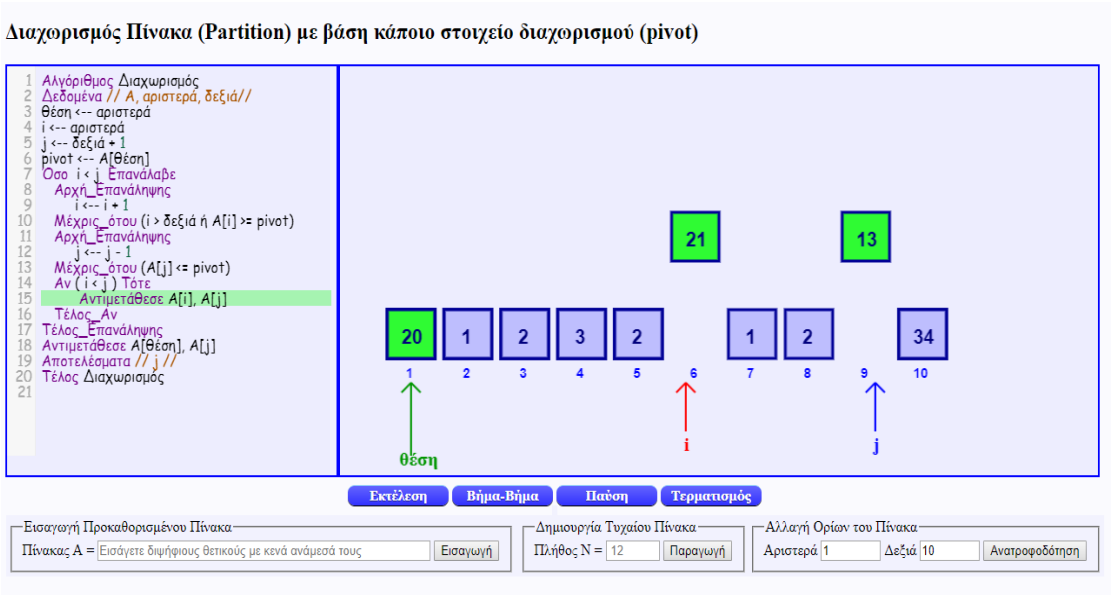
πίνακα σε δύο μέρη με βάση την τιμή ενός στοιχείου που έχει επιλεγεί, έτσι ώστε τα μικρότερα στοιχεία από αυτό να βρεθούν αριστερά και τα μεγαλύτερα δεξιά. Έτσι η θέση που θα τοποθετηθεί το στοιχείο είναι μεταξύ των δύο αυτών τμημάτων.

```
Αλγόριθμος Διαχωρισμός
Δεδομένα // A, αριστερά, δεξιά //
    θέση ← αριστερά
    i ← αριστερά
    j ← δεξιά + 1
    pivot ← A[ θέση ]
    Όσο  i < j  Επανάλαβε
        Αρχή_Επανάληψης
            i ← i + 1
        Μέχρις_ότου  i > δεξιά  ή  A[ i ] ≥ pivot
        Αρχή_Επανάληψης
            j ← j – 1
        Μέχρις_ότου  A[ j ] ≤ pivot
        Αν  i < j  Τότε
            Αντιμετάθεσε A[ i ], A[ j ]
        Τέλος_Αν
    Τέλος_Επανάληψης
    Αντιμετάθεσε A[ θέση ], A[ j ]
Αποτελέσματα // j //
Τέλος Διαχωρισμός
```

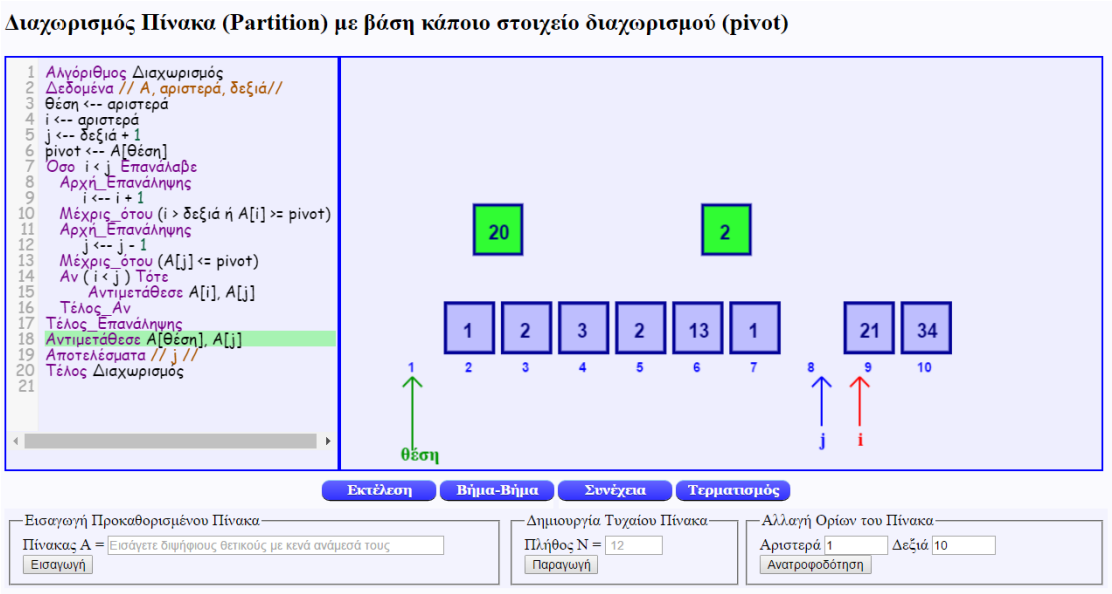
**Αλγόριθμος 7.7. Διαχωρισμός με βάση κάποιο στοιχείο (pivot)**

Στο παράδειγμα του Σχήματος 7.21 το στοιχείο που έχει επιλεγεί είναι το 20. Όταν βρεθούν δυο στοιχεία, που θα έπρεπε να βρίσκονται εκατέρωθεν του 20, τα οποία βρίσκονται σε λάθος θέσεις (21 και 13), τότε αυτά αλλάζουν αμοιβαία θέση.

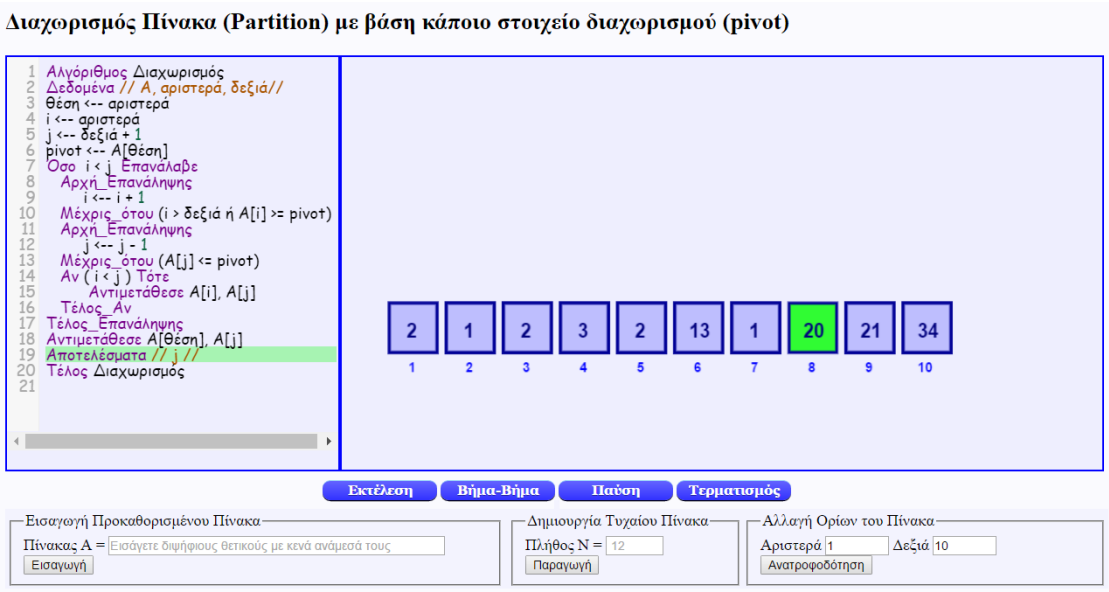
Στο Σχήμα 7.22 το στοιχείο 20 τοποθετείται στη σωστή θέση (8) έτσι ώστε όλα τα στοιχεία που είναι πριν από αυτό να είναι μικρότερά του και όλα όσα βρίσκονται μετά από αυτό να είναι μεγαλύτερά του.




**Σχήμα 7.21. Αλγόριθμος Διαχωρισμού στο DAVE**




**Σχήμα 7.22. Αλγόριθμος Διαχωρισμού: Αντιμετάθεση στοιχείων στο DAVE**




**Σχήμα 7.23. Αλγόριθμος Διαχωρισμού: Τοποθέτηση στοιχείου στη σωστή θέση**

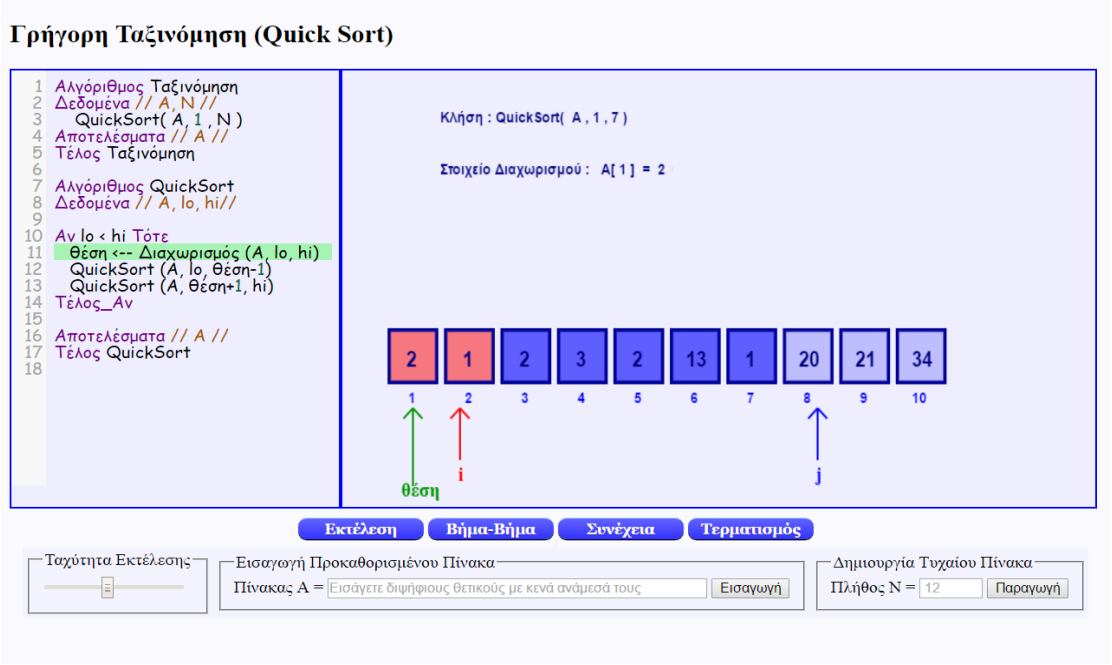
### *7.6.6 Αλγόριθμος γρήγορης ταξινόμησης*

Ο αλγόριθμος της γρήγορης ταξινόμησης (quicksort) θεωρείται πρακτικά ο πιο γρήγορος μεταξύ των γνωστών αλγορίθμων , παρόλο που η πολυπλοκότητά του στη χειρότερη περίπτωση είναι $O(n^2)$. Ο λόγος είναι ότι στη μέση περίπτωση έχει πολυπλοκότητα O(nlogn) (Knuth 1998). Η αναδρομική φύση του αλγορίθμου καθιστά δύσκολη την προσέγγισή του από τους μαθητές, και για το λόγο αυτό δεν διδάσκεται στην δευτεροβάθμια εκπαίδευση. Λόγω της σπουδαιότητάς του αποφασίστηκε να ενταχθεί και ο αλγόριθμος γρήγορης ταξινόμησης στο DAVE.

Σε κάθε βήμα αναγράφεται ψηλά το στοιχείο διαχωρισμού και η αναδρομική κλήση που εκτελείται αυτή τη στιγμή. Το τμήμα του πίνακα που επεξεργάζεται η τρέχουσα αναδρομική κλήση επισημαίνεται με μπλε κλειστό χρώμα όπως φαίνεται παραπάνω.

```
Αλγόριθμος ΓρήγορηΤαξινόμηση
Δεδομένα // A, αριστερά, δεξιά //
    Αν αριστερά < δεξιά Τότε
        θέση ← Διαχωρισμός( A, αριστερά, δεξιά )
        ΓρήγορηΤαξινόμηση ( A, αριστερά, θέση–1 )
        ΓρήγορηΤαξινόμηση ( A, θέση+1, δεξιά )
    Τέλος_Αν
Αποτελέσματα // A //
Τέλος ΓρήγορηΤαξινόμηση
```

**Αλγόριθμος 7.8. Γρήγορη ταξινόμηση (QuickSort).**



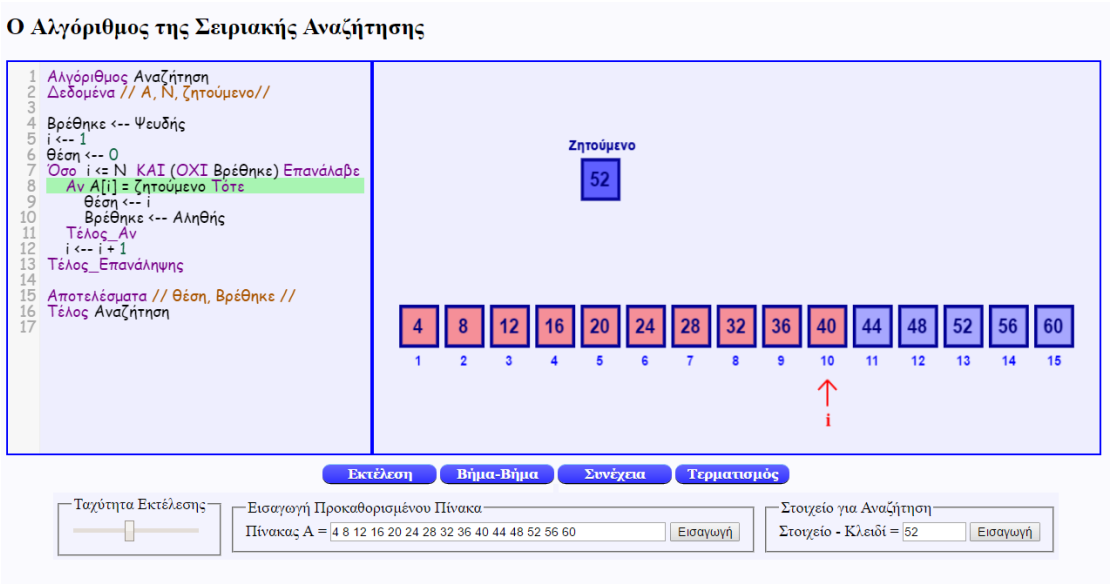

**Σχήμα 7.24. Αλγόριθμος γρήγορης ταξινόμησης στο DAVE.**

### *7.6.7 Αλγόριθμος σειριακής αναζήτησης*

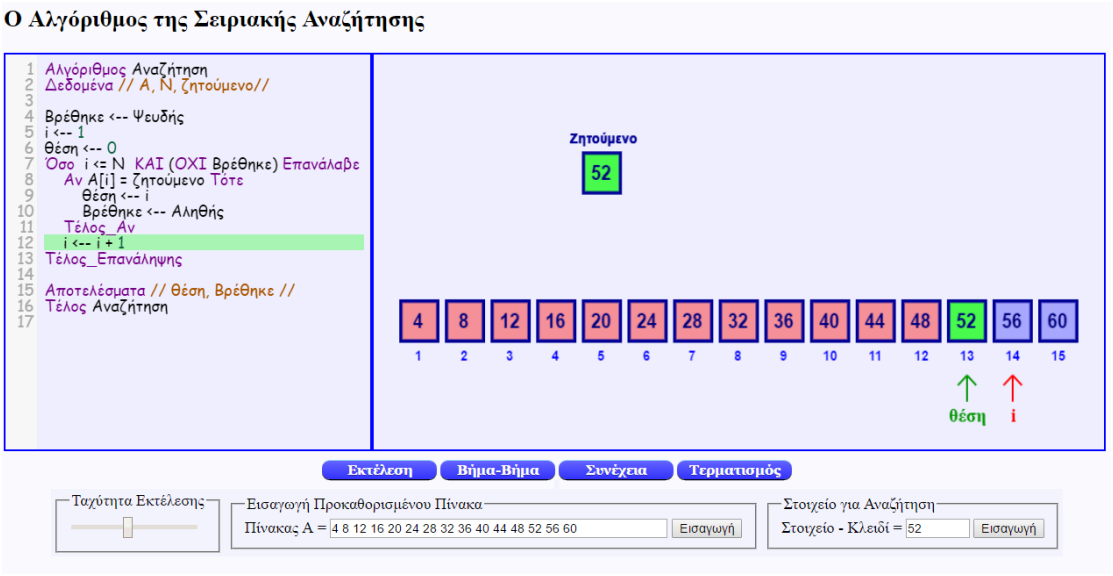
Ο αλγόριθμος της σειριακής (sequential) ή γραμμικής (linear) αναζήτησης είναι ο πιο απλός αλγόριθμος αναζήτησης.

```
Αλγόριθμος ΣειριακήΑναζήτηση
Δεδομένα // A, N, ζητούμενο //
    βρέθηκε ← Ψευδής
    i ← 1
    θέση ← 0
    Όσο i ≤ N και όχι βρέθηκε Επανάλαβε
        Αν A[ i ] = ζητούμενο Τότε
            θέση ← i
            βρέθηκε ← Αληθής
        Τέλος_Αν
        i ← i + 1
    Τέλος_Επανάληψης
Αποτελέσματα // θέση, βρέθηκε //
Τέλος ΣειριακήΑναζήτηση
```

**Αλγόριθμος 7.9. Σειριακή Αναζήτηση (Linear Search)**




**Σχήμα 7.25. Αλγόριθμος Σειριακής Αναζήτησης στο DAVE**




**Σχήμα 7.26. Αλγόριθμος Σειριακής Αναζήτησης : Εύρεση του στοιχείου στο DAVE**

Σύμφωνα με αυτόν όταν ψάχνουμε ένα στοιχείο ελέγχουμε εξαντλητικά όλα τα στοιχεία του πίνακα μέχρι να το βρούμε ή να διαπιστώσουμε ότι δεν υπάρχει. Στο Σχήμα 7.25 εκτελείται αναζήτηση του στοιχείου 52 στον πίνακα Α.

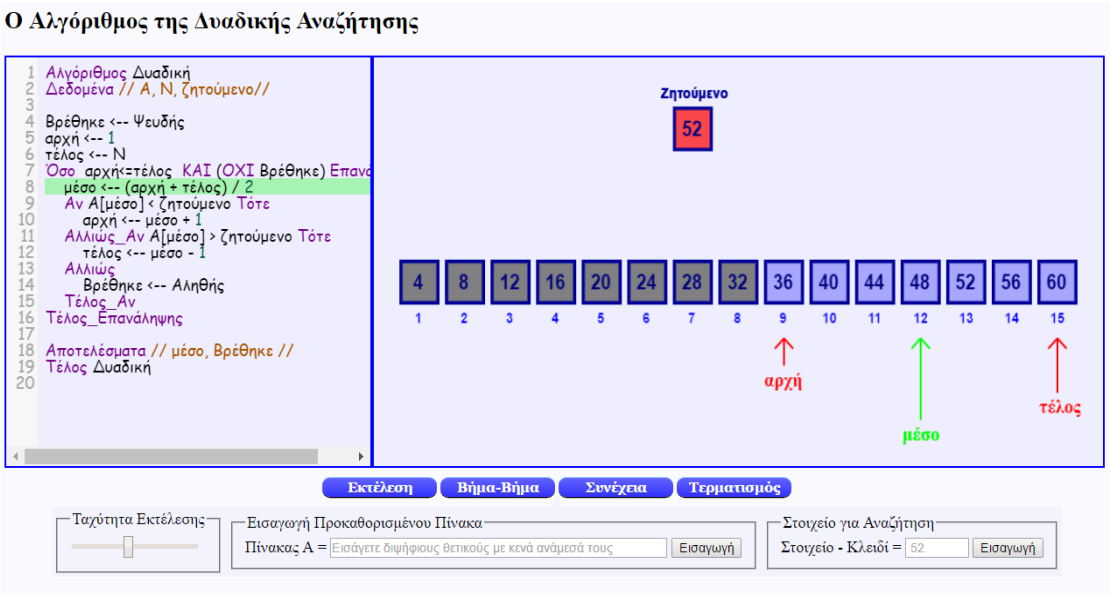
Στο Σχήμα 7.26 ο αλγόριθμος έχει βρει το στοιχείο στη θέση 13. Αν το στοιχείο δεν υπάρχει στον πίνακα, τότε ο αλγόριθμος θα ελέγξει όλα τα στοιχεία του πίνακα και για αυτό έχει γραμμική πολυπλοκότητα O(n).

### *7.6.8 Αλγόριθμος δυαδικής αναζήτησης*

Στην περίπτωση που τα στοιχεία του πίνακα είναι ταξινομημένα τότε υπάρχει ένας τρόπος να αξιοποιηθεί αυτή τη διάταξη για να βρεθεί πολύ γρήγορα το ζητούμενο στοιχείο. Αρχικά συγκρίνεται το μεσαίο στοιχείο με το ζητούμενο. Αν το ζητούμενο είναι μεγαλύτερο θα βρίσκεται δεξιά, ενώ αν είναι μικρότερο αριστερά. Σε κάθε περίπτωση έχει αποφευχθεί η σύγκριση των μισών στοιχείων του πίνακα. Με το ίδιο σκεπτικό το πρόβλημα διαιρείται συνεχώς στα δύο μέχρι να βρούμε το ζητούμενο. Ο αλγόριθμος αυτός είναι γνωστός ως δυαδική αναζήτηση (binary search) και έχει λογαριθμική πολυπλοκότητα O(logn). Αυτό σημαίνει ότι αν κάποιος μας δώσει μια λίστα με 1 δισεκατομμύριο $\approx 2^{30}$ ονόματα χρειαζόμαστε μόνο $\log 2^{30} = 30$ συγκρίσεις!

```
Αλγόριθμος ΔυαδικήΑναζήτηση
Δεδομένα // A, N, ζητούμενο //
    βρέθηκε ← Ψευδής
    lo ← 1
    hi ← N
    Όσο lo ≤ hi και όχι Βρέθηκε Επανάλαβε
        μέσο ← ( lo + hi ) / 2
        Αν A[ μέσο ] < ζητούμενο Τότε
            lo ← μέσο + 1
        Αλλιώς_Αν A[ μέσο ] > ζητούμενο Τότε
            hi ← μέσο – 1
        Αλλιώς
            Βρέθηκε ← Αληθής
        Τέλος_Αν
        i ← i + 1
    Τέλος_Επανάληψης
Αποτελέσματα // μέσο, Βρέθηκε //
Τέλος ΔυαδικήΑναζήτηση
```

**Αλγόριθμος 7.10. Δυαδική Αναζήτηση (Binary Search)**




**Σχήμα 7.27. Αλγόριθμος Δυαδικής Αναζήτησης στο DAVE**

Η οπτικοποίηση της δυαδικής αναζήτησης διαφέρει από τις οπτικοποιήσεις άλλων αλγορίθμων όσον αφορά τον χρωματισμό των στοιχείων του πίνακα που έχουν αποκλειστεί από την αναζήτηση με ένα γκρίζο χρώμα. Αυτό δείχνει οπτικά πόσο μικραίνει ο χώρος αναζήτησης σε κάθε βήμα. Στο σχήμα 7.27 εκτελείται αναζήτηση του αριθμού 52. Μετά τη σύγκριση με το στοιχείο 32 στη θέση 8 και τη διαπίστωση ότι το 52 βρίσκεται δεξιά αφού 52>32 ο χώρος αναζήτησης περιορίζεται στο τμήμα του πίνακα από τις θέσεις 9 έως και 15. Στη συνέχεια γίνεται σύγκριση με το 48 που είναι στη μέση του νέου τμήματος κ.ο.κ.

### *7.6.9 Πρόσθετοι Αλγόριθμοι*

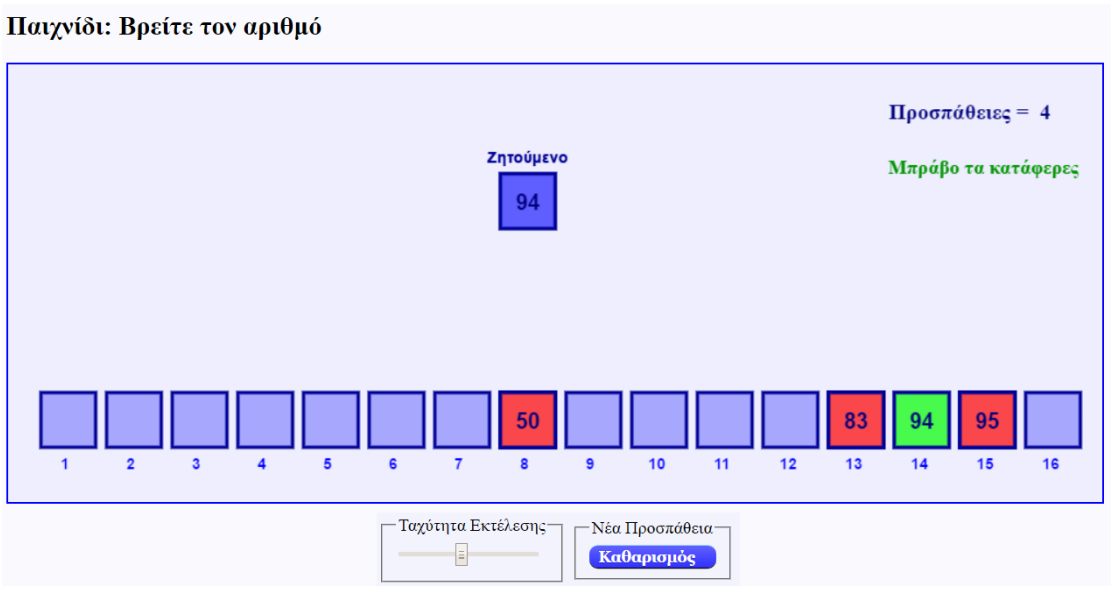
Το σύστημα εξελίσσεται συνεχώς και προστίθενται νέα χαρακτηριστικά και αλγόριθμοι.

Ένα από αυτά είναι το παιχνίδι "*Βρες τον αριθμό*" σε μονοδιάστατο αλλά και δισδιάστατο πίνακα όπως φαίνεται στα Σχήματα 7.28, 7.29, 7.30, 7.31. Το παρακάτω παιχνίδι μπορεί να χρησιμοποιηθεί για την εισαγωγή των μαθητών στην δυαδική αναζήτηση, καθοδηγώντας τους να "επινοήσουν" τη βασική ιδέα της μέσα από αυτό.

Τα παρακάτω παιχνίδια αναζήτησης θα μπορούσαν να αποτελέσουν την κεντρική ιδέα δραστηριοτήτων υπολογιστικής σκέψης σχετικά με τον σχεδιασμό αλγόριθμων αναζήτησης από τους μαθητές, έτσι ώστε να αξιοποιήσουν τα ιδιαίτερα χαρακτηριστικά του συγκεκριμένου χώρου αναζήτησης.

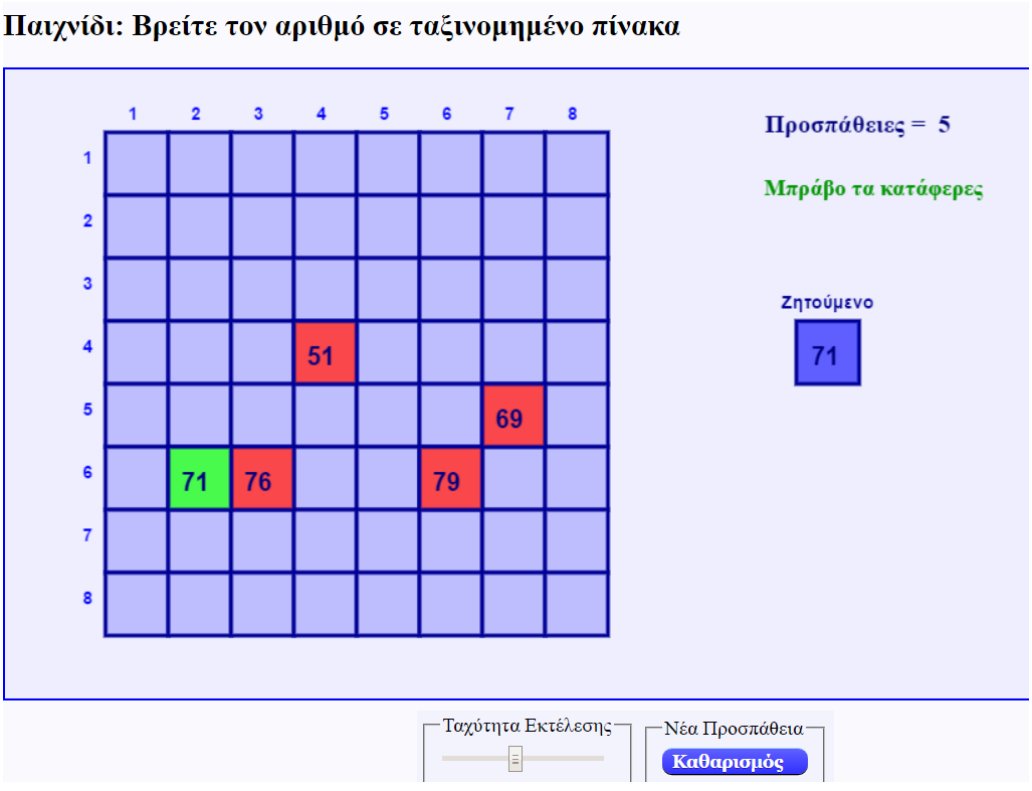
Στο παράδειγμα με τον πίνακα δυο διαστάσεων οι μαθητές δεν αξιοποίησαν μόνο την αύξουσα σειρά των στοιχείων του πίνακα αλλά και το γεγονός ότι όλοι οι αριθμοί είναι από 1 μέχρι 99, σχεδιάζοντας έναν αλγόριθμο δυαδικής αναζήτησης ειδικά για αυτή την περίπτωση.


**Σχήμα 7.28. Παιχνίδι αναζήτησης σε μονοδιάστατο πίνακα**




**Σχήμα 7.29. Παιχνίδι αναζήτησης σε μονοδιάστατο πίνακα στο DAVE.**


**Σχήμα 7.30. Παιχνίδι αναζήτησης σε πίνακα δύο διαστάσεων στο DAVE.**




**Σχήμα 7.31. Παιχνίδι αναζήτησης σε πίνακα δύο διαστάσεων στο DAVE.**

# 8

# *Μελέτη της Συμβολής του Περιβάλλοντος DAVE*

## *8.1 Σκοπός και Ερευνητικά ερωτήματα*

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της έρευνας για τη μελέτη της συμβολής του DAVE στην κατανόηση της λειτουργίας των αλγορίθμων ταξινόμησης από μαθητές Λυκείου.

Ο σκοπός της έρευνας ήταν να διερευνηθεί κατά πόσο το περιβάλλον δυναμικής οπτικοποίησης αλγορίθμων DAVE και τα χαρακτηριστικά που ενσωματώνει συμβάλλουν στη διερεύνηση της λειτουργίας του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής και στην οικοδόμηση της λογικής του από τους μαθητές της Γ' Λυκείου που παρακολουθούν το μάθημα "Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον". Τα ερευνητικά ερωτήματα που τέθηκαν ήταν τα εξής:

- Σε ποιο βαθμό το λογισμικό οπτικοποίησης DAVE ενισχύει την εμπλοκή των μαθητών στην επίλυση αλγοριθμικών προβλημάτων ταξινόμησης;
- Συμβάλλει το λογισμικό οπτικοποίησης DAVE στην οικοδόμηση επαρκών αναπαραστάσεων για τη λειτουργία των αλγορίθμων ταξινόμησης;
- Σε ποιο βαθμό βοηθήθηκαν οι συμμετέχοντες μαθητές να βελτιώσουν τις προτεινόμενες λύσεις τους μετά τη χρήση του λογισμικού;
- Ποιες δυνατότητες του περιβάλλοντος αξιοποίησαν οι μαθητές και ποιες πρακτικές επίλυσης υιοθέτησαν κατά την ενασχόλησή τους με την οπτικοποίηση αλγορίθμων;

## *8.2 Μεθοδολογία*

Η έρευνα έλαβε χώρα, σε δύο γενικά λύκεια του δήμου Περιστερίου Αττικής. Το δείγμα περιελάβανε 45 μαθητές της Γ' τάξης της τεχνολογικής κατεύθυνσης που παρακολουθούσαν το μάθημα Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον (ΑΕΠΠ). Δεν υπήρξε καμία διδακτική παρέμβαση πριν τη διεξαγωγή της έρευνας. Οι μαθητές όλων των τμημάτων είχαν διδαχθεί την ταξινόμηση ευθείας ανταλλαγής.

Η έρευνα διεξήχθη στο εργαστήριο πληροφορικής. Οι μαθητές χωρίστηκαν σε ομάδες των 11-12 μαθητών, έτσι ώστε να αντιστοιχεί ένας Η/Υ ανά μαθητή. Διατέθηκαν δύο συνεχόμενες διδακτικές ώρες, κατά τις οποίες ήταν παρών και ο καθηγητής πληροφορικής κάθε τμήματος.

Αρχικά, οι μαθητές απάντησαν σε μια σειρά από αλγοριθμικά προβλήματα σχετικά με τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής στο τετράδιό τους χωρίς τη χρήση του λογισμικού. Στη συνέχεια ζητήθηκε να χρησιμοποιήσουν το λογισμικό για να διορθώσουν ή να σχεδιάσουν εκ νέου τις αρχικές λύσεις τους. Προηγήθηκε μια δεκάλεπτη παρουσίαση του λογισμικού από τον ερευνητή στους μαθητές.

Στο τέλος της έρευνας ζητήθηκε από τους μαθητές να απαντήσουν σε ένα σύντομο ερωτηματολόγιο σχετικά με το λογισμικό. Οι μαθητές μπορούσαν να σχολιάσουν γενικά το λογισμικό, να αναφέρουν πλεονεκτήματα και προβλήματα που διαπίστωσαν ή ακόμα και ενδεχόμενες επεκτάσεις.

## *8.3 Εργαλείο της έρευνας*

Οι μαθητές κλήθηκαν να απαντήσουν σε μια σειρά από ερωτήσεις και να επιλύσουν προβλήματα ταξινόμησης πινάκων, έχοντας τη δυνατότητα να χρησιμοποιήσουν παράλληλα το εκπαιδευτικό περιβάλλον προγραμματισμού DAVE. Για την υλοποίηση των δραστηριοτήτων χρησιμοποιήθηκε ειδικό φύλλο εργασίας, το οποίο σχεδιάστηκε στο πλαίσιο της παρούσας μελέτης. Το φύλλο εργασίας ήταν επώνυμο, ώστε να έχουμε τη δυνατότητα στη συνέχεια να διερευνήσουμε περαιτέρω τις απόψεις των μαθητών που θα είχαν ερευνητικό ενδιαφέρον, μέσω συνεντεύξεων. Από τους μαθητές ζητήθηκε να παραθέσουν τις ιδέες και τις προσεγγίσεις τους σχετικά με τις δραστηριότητες του φύλλου εργασίας, όσο πιο αναλυτικά μπορούσαν.

Η πρώτη δραστηριότητα που δόθηκε ήταν δομημένη σε βήματα βαθμιαίας αυξανόμενης δυσκολίας, έτσι ώστε να βοηθήσει τους μαθητές να εξοικειωθούν με το λογισμικό.

## *8.4 Ανάλυση των αποτελεσμάτων*

Στην παράγραφο αυτή παρουσιάζεται η ανάλυση των απαντήσεων που δόθηκαν από τους μαθητές σε τρεις ενδεικτικές δραστηριότητες-προβλήματα, με τα οποία ασχολήθηκαν οι μαθητές του δείγματος, καθώς και τα σημαντικότερα αποτελέσματα.

### *8.4.1 Δραστηριότητα 1*

Η πρώτη δραστηριότητα αποτελείται από 6 βήματα (μικρές δραστηριότητες) με διττό σκοπό. Αφενός, να εξοικειωθούν οι μαθητές με το λογισμικό και, αφετέρου, να ελεγχθούν κάποιες βασικές γνώσεις των μαθητών πάνω στον αλγόριθμο ταξινόμησης, χρησιμοποιώντας απλές ερωτήσεις ώστε να ενισχυθεί η αυτοπεποίθησή τους και να ενθαρρυνθεί η συμμετοχή τους. Τα δύο πρώτα ερωτήματα ήταν απλά ερωτήματα εκτέλεσης του αλγορίθμου στα οποία απάντησαν σχεδόν όλοι οι μαθητές.

Στη συνέχεια αναλύουμε τα τέσσερα τελευταία ερωτήματα της δραστηριότητας, με τα οποία ζητήθηκε από τους μαθητές να τροποποιήσουν τον αλγόριθμο που τους δόθηκε.

**Ερώτημα 3**

**Για** i **από** 2 **μέχρι** 2
  **Για** j **από** N **μέχρι** i **με βήμα** –1
    **Αν** A[ j ] < A[ j – 1 ] **Τότε**
      *Αντιμετάθεσε* A[ j ], A[ j – 1 ]
    **Τέλος_Αν**
  **Τέλος_Επανάληψης**
**Τέλος_Επανάληψης**

*Να περιγράψετε τη λειτουργία που επιτελεί το διπλανό τμήμα κώδικα.*

Το παραπάνω τμήμα κώδικα εκτελεί ένα μόνο πέρασμα στα στοιχεία του πίνακα, οπότε το αναμενόμενο αποτέλεσμα είναι η τοποθέτηση του ελάχιστου στοιχείου στην πρώτη θέση (A[1]), μετά από διαδοχικές αντιμεταθέσεις. Παρόλα αυτά μόνο 18 μαθητές απάντησαν σωστά, ενώ 23 έδωσαν την απάντηση ότι ο αλγόριθμος ταξινομεί όλο τον πίνακα. Επίσης τέσσερις (4) μαθητές έδωσαν απαντήσεις στις οποίες περιγράφουν τον κώδικα γραμμή-γραμμή. Μια τέτοια χαρακτηριστική απάντηση δίνεται παρακάτω:

*«Για κάθε i και j ο αλγόριθμος συγκρίνει τα στοιχεία A[j], A[j-1], και αν A[j]<A[j-1] τότε αυτά αλλάζουν θέση.»*

Οι απαντήσεις αυτές κατατάσσονται στο πολυδομικό επίπεδο της ταξινομίας SOLO αφού οι μαθητές δεν έχουν διακρίνει τη λογική του αλγορίθμου και τη λειτουργία που επιτελεί.

Στη συνέχεια ζητήθηκε από τους μαθητές να δοκιμάσουν τον παραπάνω αλγόριθμο στο λογισμικό οπτικοποίησης και να αναδιατυπώσουν ή να θεωρήσουν τις απαντήσεις τους. Συνολικά 9 μαθητές, οι οποίοι είχαν απαντήσει αρχικά ότι ο αλγόριθμος εκτελεί ταξινόμηση όλου του πίνακα, διόρθωσαν τις απαντήσεις τους αφού εκτέλεσαν την οπτικοποίηση του αλγορίθμου. Μετά τη χρήση του εργαλείου όμως έδωσαν απαντήσεις όπως οι παρακάτω:

*«Βρίσκει το μικρότερο στοιχείο του πίνακα.»*
*«Ο αλγόριθμος βρίσκει το min.»*
*«Ο αλγόριθμος βάζει το μικρότερο στην αρχή.»*

Ωστόσο 22 από τους μαθητές επέλεξαν να μην χρησιμοποιήσουν το εργαλείο για αυτή την άσκηση γιατί θεώρησαν ότι δεν το χρειάζονται. Τα συγκεντρωτικά αποτελέσματα φαίνονται στον Πίνακα 8.1.

**Πίνακας 8.1. Απαντήσεις μαθητών στο βήμα 3 της Δραστηριότητας 1**

| **Απαντήσεις (N = 45)** | **Στο χαρτί** | **Με χρήση DAVE** |
|---|---|---|
| περιγραφή γραμμή προς γραμμή | 4 | 3 |
| εκτελεί ταξινόμηση | 23 | 15 |
| θέτει το ελάχιστο στοιχείο στην πρώτη θέση | 18 | 27 |

**Ερώτημα 4 :** *Να τροποποιήσετε τον αλγόριθμο έτσι ώστε στην πρώτη θέση του πίνακα να μεταφερθεί ο μεγαλύτερος αριθμός. Να αιτιολογήσετε την απάντησή σας.*

Σαράντα (40) μαθητές έδωσαν σωστή απάντηση τροποποιώντας το συγκριτικό τελεστή από '<' σε '>'. Μόνο 6 από αυτούς αιτιολόγησαν την απάντησή τους γράφοντας

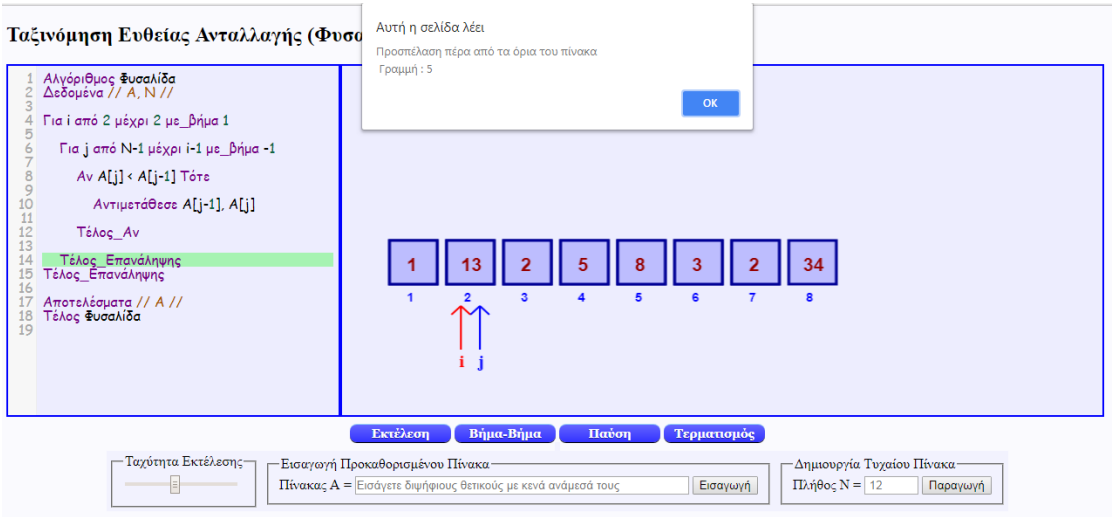
*«πρέπει ο επόμενος να είναι μεγαλύτερος από τον προηγούμενο».*

Ενδιαφέρον παρουσιάζει το γεγονός ότι 2 μαθητές δεν άλλαξαν τη φορά της ανίσωσης αλλά τους δείκτες των στοιχείων του πίνακα αλλάζοντας τη συνθήκη από

A[j] < A[j–1] σε A[j] < A[j+1]. Αυτό, στη γενική περίπτωση είναι σωστό, αλλά στις οριακές τιμές έχουμε προσπέλαση πέρα από τα όρια του πίνακα, όπως φαίνεται στο Τμήμα Κώδικα 8.1. Οι μαθητές εντόπισαν το λάθος, αφού το DAVE εμφανίζει μήνυμα για αυτή την περίπτωση. Στη συνέχεια έδωσαν το Τμήμα Κώδικα 8.2 το οποίο όμως πάλι εκτελεί προσπέλαση πέρα από τα όρια του πίνακα όπως φαίνεται στο Σχήμα 8.1. Επίσης δεν ανεβάζει στην πρώτη θέση το μεγαλύτερο αλλά το μικρότερο. Οι μαθητές εντόπισαν αυτά τα προβλήματα, διόρθωσαν τα όρια της επαναληπτικής δομής και άλλαξαν το συγκριτικό τελεστή από '<' σε '>' έτσι ώστε να ανεβαίνει το μεγαλύτερο στοιχείο στην πρώτη θέση, όπως φαίνεται στο Τμήμα Κώδικα 8.3.

```
Για i από 2 μέχρι 2
    Για j από N μέχρι i με βήμα -1
        Αν A[ j ] < A[ j+1] Τότε Αντιμετάθεσε A[ j ], A[ j+1]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.1. Προσπέλαση πέρα από τα όρια του πίνακα**



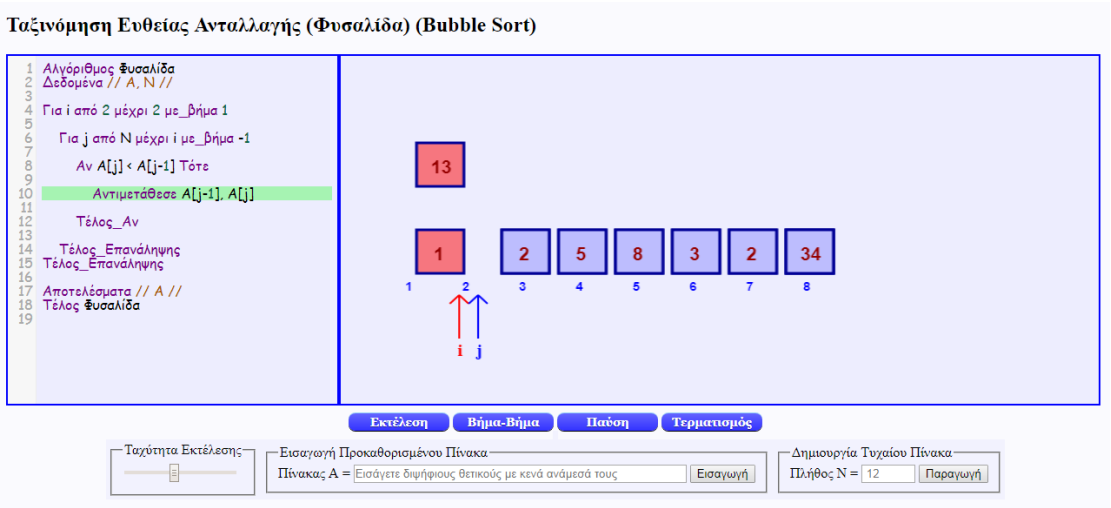

**Σχήμα 8.1. Διαγνωστικό μήνυμα για την προσπέλαση πέρα από τα όρια του πίνακα**

```
Για i από 2 μέχρι 2
    Για j από N–1 μέχρι i–1 με βήμα -1
        Αν A[ j ] < A[ j-1] Τότε Αντιμετάθεσε A[ j ], A[ j-1]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.2. Προσπέλαση πέρα από τα όρια του πίνακα**

```
Για i από 2 μέχρι 2
    Για j από N μέχρι i με βήμα -1
        Αν A[ j ] > A[ j–1] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.3. Προσπέλαση πέρα από τα όρια του πίνακα**




**Σχήμα 8.2. Ο αριθμός 1 ανεβαίνει στην πρώτη θέση του πίνακα**

**Πίνακας 8.2. Απαντήσεις μαθητών στο βήμα 4 της Δραστηριότητας 1**

| Απαντήσεις (N = 45) | Στο χαρτί | Με χρήση DAVE |
|---|---|---|
| καμία απάντηση | 3 | 3 |
| αλλαγή του συγκριτικού τελεστή | 38 | 38 |
| A[j]<A[j+1] με σωστά όρια στη δομή επανάληψης | 0 | 4 |
| A[j]<A[j+1] με προσπέλαση πέρα από τα όρια του πίνακα | 4 | 0 |

Η οπτικοποίηση του αλγορίθμου και ειδικά των δεικτών i, j βοήθησε τους τέσσερις μαθητές να εντοπίσουν και να διορθώσουν τον αλγόριθμό τους ώστε να μην γίνεται προσπέλαση πέρα από τα όρια του πίνακα. Επίσης μέσα από την οπτικοποίηση του αλγορίθμου παρατήρησαν αν ανεβαίνει το μικρότερο ή το μεγαλύτερο στοιχείο ώστε να τροποποιήσουν κατάλληλα τον συγκριτικό τελεστή.

**Ερώτημα 5 :** *Να τροποποιήσετε τον αλγόριθμο έτσι ώστε στις 3 πρώτες θέσεις του πίνακα να βρεθούν οι 3 μεγαλύτεροι αριθμοί. Να αιτιολογήσετε την απάντησή σας.*

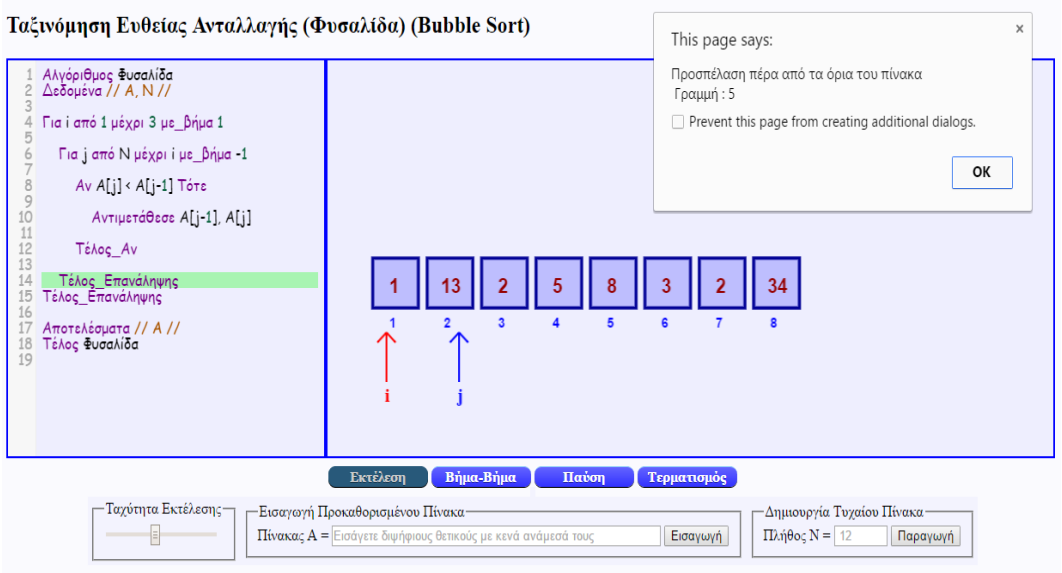
Στο ερώτημα αυτό 14 μαθητές έδωσαν σωστή απάντηση τροποποιώντας την εξωτερική επανάληψη (*Για i από 2 μέχρι 4*), ώστε ο αλγόριθμος να εκτελεί μόνο τρία περάσματα. Δέκα μαθητές έδωσαν πλήρη αλγόριθμο ταξινόμησης, ο οποίος ταξινομεί όλα τα στοιχεία και όχι μόνο τα 3 πρώτα (*Για i από 2 μέχρι N*). Οκτώ από τους δέκα μαθητές αναθεώρησαν την απάντησή τους αφού πειραματίστηκαν με την οπτικοποίηση του αλγορίθμου, ώστε να εκτελεί μόνο τρία περάσματα.

Επίσης, έξι μαθητές έδωσαν μια απάντηση στην οποία εκτελούνται τέσσερα περάσματα (*Για i από 2 μέχρι 5*). Αφού παρακολούθησαν την οπτικοποίηση του αλγορίθμου πέντε από τους έξι μαθητές παρατήρησαν ότι μετά το τρίτο πέρασμα τα στοιχεία είναι ήδη ταξινομημένα, άρα δεν χρειάζεται τέταρτο πέρασμα, οπότε διόρθωσαν το άνω άκρο της επανάληψης από 5 σε 4.

Άλλοι 4 μαθητές έδωσαν μια αποδοτική λύση από την οποία φαίνεται ότι κατάλαβαν πως αρκούν τρία περάσματα, όμως η λύση τους δεν ήταν σωστή (*Για i από 1 μέχρι 3*), γιατί για j=i=1 έχουμε προσπέλαση στο στοιχείο A[j-1] = A[i-1] = A[1-1] = A[0], το οποίο είναι πέρα από τα όρια του πίνακα. Αυτό το λάθος μπορεί να φανεί εποπτικά από τον μαθητή κατά την οπτικοποίηση, όμως εντοπίζεται και από το εργαλείο το οποίο εμφανίζει κατάλληλο διαγνωστικό μήνυμα σε ένα αναδυόμενο παράθυρο. Συνολικά, 18 από τους 31 μαθητές αναθεώρησαν τις απαντήσεις μετά από την αλληλεπίδρασή τους με το λογισμικό.

**Πίνακας 8.3. Απαντήσεις μαθητών στο βήμα 5 της Δραστηριότητας 1**

| **Απαντήσεις (N = 45)** | **Στο χαρτί** | **Με χρήση DAVE** |
|---|---|---|
| ημιτελείς απαντήσεις | 3 | 3 |
| για i=2 μέχρι 4 | 14 | 32 |
| για i=1 μέχρι 3 | 7 | 2 |
| για i=2 μέχρι 5 | 6 | 1 |
| για i=2 μέχρι N | 15 | 7 |

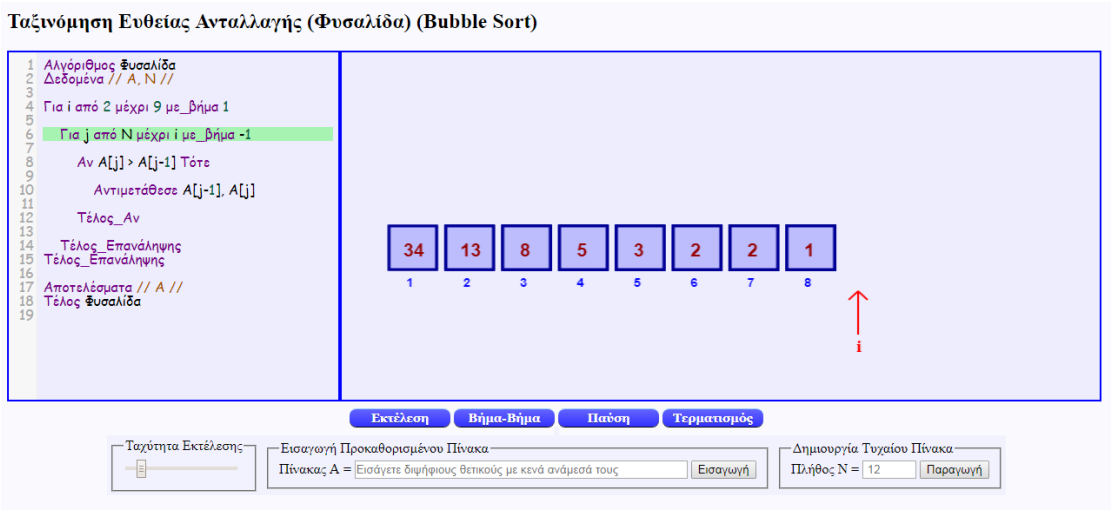



**Σχήμα 8.3. Το λογισμικό εντοπίζει την προσπέλαση πέρα από τα όρια του πίνακα**

**Ερώτημα 6:** *Τι πρέπει να αλλάξετε στον αλγόριθμο έτσι ώστε να ταξινομηθούν όλα τα στοιχεία του πίνακα σε φθίνουσα σειρά;*

Οι μαθητές εργάστηκαν πάνω στον αλγόριθμο που είχαν ήδη αναπτύξει στο προηγούμενο ερώτημα, οπότε αρκούσε να αναπροσαρμόσουν τα όρια της εξωτερικής δομής επανάληψης ώστε να ταξινομηθούν όλα τα στοιχεία του πίνακα και όχι μόνο τα τρία πρώτα. Την απάντηση (*Για i από 2 μέχρι 8*) έδωσαν 28 μαθητές ενώ 7 απάντησαν ότι "*πρέπει να εκτελεστεί 7 φορές*" χωρίς να διατυπώσουν κάποιον αλγόριθμο. Τρεις μαθητές έδωσαν την απάντηση "*ο αλγόριθμος πρέπει να εκτελεστεί 9 φορές*". Όταν ζητήθηκε να διευκρινίσουν αυτό που εννοούσαν, έδωσαν το τμήμα κώδικα που εκτελεί την ταξινόμηση αλλά κάνει μια επανάληψη παραπάνω (*Για i από 2 μέχρι 9*). Εδώ αναδείχθηκε η παρανόηση ότι η συγκεκριμένη επανάληψη εκτελείται 9 φορές, επιβεβαιώνοντας τα ευρήματα άλλων ερευνών (Dale 2006· du Boulay, 1986· Robins et al., 2003) ότι οι παρανοήσεις στις δομές επανάληψης συνιστούν παράγοντα δυσκολιών στην κατανόηση πιο σύνθετων δομών.

Όταν οι μαθητές αυτοί πειραματιστήκαν με το λογισμικό εντόπισαν ότι υπάρχει προσπέλαση πέρα από τα όρια του πίνακα από τη θέση του δείκτη i ο οποίος προχώρησε πέρα από το 8ο στοιχείο του πίνακα. Αυτή η οπτικοποίηση τους βοήθησε να διορθώσουν τον αλγόριθμο ώστε ο δείκτης i να σταματάει όταν φτάσει στον αριθμό 8, όπως φαίνεται στο Τμήμα Κώδικα 8.4.




**Σχήμα 8.4. Ο δείκτης i έχει μετακινηθεί πέρα από τα όρια του πίνακα**

```
Για i από 2 μέχρι 8
    Για j από N μέχρι i με βήμα -1
        Αν A[ j ] > A[ j–1] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.4. Διόρθωση του άνω ορίου της εξωτερικής επανάληψης**

**Πίνακας 8.4.Απαντήσεις μαθητών στο βήμα 6 της Δραστηριότητας 1**

| Απαντήσεις (N = 45) | Στο χαρτί | Με χρήση DAVE |
|---|---|---|
| Δεν απάντησαν | 3 | 3 |
| για i=2 μέχρι 8 | 28 | 35 |
| χρειάζονται 7 περάσματα του αλγορίθμου | 7 | 0 |
| για i=2 μέχρι 9 | 7 | 0 |

Οι μαθητές που απάντησαν ότι απαιτούνται 7 περάσματα δεν χρησιμοποιήσαν το λογισμικό με το επιχείρημα ότι είναι σίγουροι/ες για την απάντησή τους και άρα δεν χρειάζεται επαλήθευση.

### *8.4.2 Δραστηριότητα 2*

*Στο πεδίο "Εισαγωγή Προκαθορισμένου Πίνακα" δώστε τους παρακάτω 6 αριθμούς χωρισμένους με ένα κενό: 10 10 10 10 10 3 και πατήστε Εισαγωγή.*

*Στη συνέχεια τροποποιήστε τον αλγόριθμο έτσι ώστε να εκτελεί αύξουσα ταξινόμηση.*

*Πόσα περάσματα θα χρειαστούν για να ταξινομηθεί ο πίνακας; Να αιτιολογήσετε την απάντησή σας*

Από τους 45 μαθητές, οι 25 απάντησαν ότι αρκούν 2 περάσματα χωρίς να χρειαστεί

**Πίνακας 8.5. Απαντήσεις μαθητών στη Δραστηριότητα 2β**

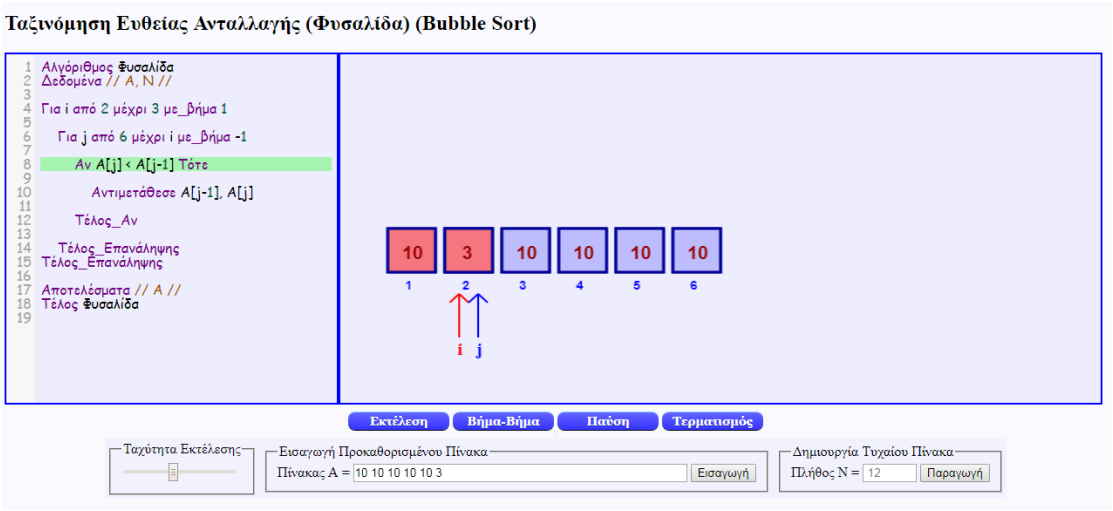
| Απαντήσεις (N = 45) | Στο χαρτί | Με χρήση DAVE |
|---|---|---|
| Αρκεί ένα πέρασμα του αλγόριθμου | 22 | 39 |
| Ταξινόμηση όλου του πίνακα | 19 | 0 |
| Τέσσερα περάσματα | 2 | 0 |
| Αντιμετάθεσε A[1], A[6] | 2 | 6 |

Πριν τη χρήση του λογισμικού, τρεις (3) μαθητές δεν έδωσαν απάντηση και 16 απάντησαν ότι χρειάζεται ταξινόμηση του πίνακα. Μετά την παρακολούθηση της οπτικοποίησης του αλγορίθμου οι μαθητές παρατήρησαν ότι μετά το πρώτο πέρασμα ο αριθμός 3 έχει ανέβει στην πρώτη θέση και ο πίνακας είναι πλέον ταξινομημένος, Μετά από αυτή την επισήμανση τροποποίησαν τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής ώστε να εκτελεί μόνο ένα πέρασμα:

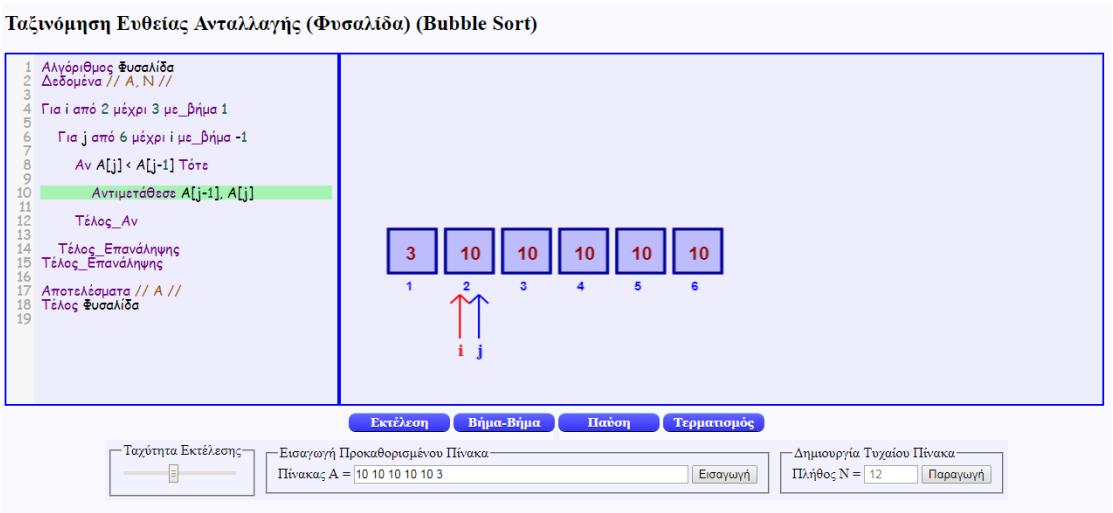
```
Για i από 2 μέχρι 2
    Για j από 6 μέχρι i με βήμα -1
        Αν A[ j ] < A[ j–1] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.5. Το μικρότερο στοιχείο ανεβαίνει στην πρώτη θέση**

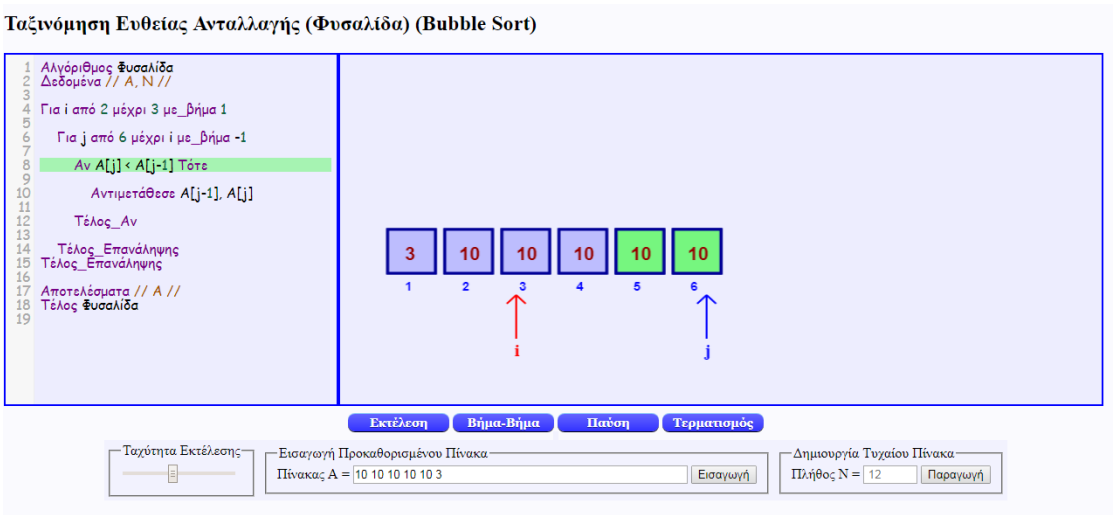
Δεν έφτασαν όμως όλοι οι μαθητές στον παραπάνω αλγόριθμο αμέσως. Πέντε μαθητές έδωσαν το Τμήμα Κώδικα 8.6 όπου εκτελούνται δυο περάσματα του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής, όπως φαίνεται στις οθόνες του λογισμικού που δίνονται στα σχήματα 8.5, 8.6 και 8.7 στο σχήμα. Ειδικά στο σχήμα 8.7 φαίνεται πλέον ότι το δεύτερο πέρασμα είναι περιττό αφού ο πίνακας είναι ήδη ταξινομημένος.




**Σχήμα 8.5. Ο αριθμός 3 ανεβαίνει στην πρώτη θέση**




**Σχήμα 8.6. Μετά το πρώτο πέρασμα τα στοιχεία του πίνακα είναι ταξινομημένα**




**Σχήμα 8.7. Τα στοιχεία A[5], A[6] βρίσκονται στη σωστή διάταξη**

Αυτό παρατήρησαν οι μαθητές που έδωσαν αυτή τη λύση με αποτέλεσμα να διορθώσουν τον αλγόριθμό τους έτσι ώστε να εκτελεί μόνο ένα πέρασμα. Αυτό αποτελεί άλλο ένα παράδειγμα για τον τρόπο με τον οποίο μπορεί να βοηθήσει η χρήση του DAVE θέματα βελτιστοποίησης αλγορίθμων.

Την απάντηση ότι αρκεί μόνο ένα πέρασμα του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής χωρίς τη χρήση του λογισμικού, έδωσαν είκοσι δυο (22) μαθητές.

```
Για i από 2 μέχρι 2
    Για j από 6 μέχρι i με βήμα -1
        Αν A[ j ] < A[ j–1] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

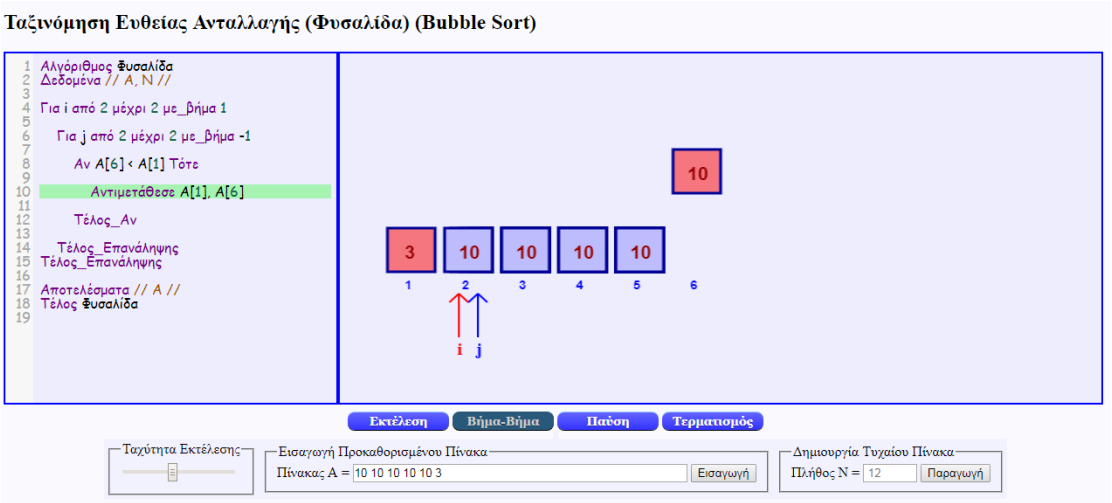
**Τμήμα Κώδικα 8.6. Αρκεί μόνο ένα πέρασμα (εξωτερική επανάληψη) του αλγορίθμου**

Συνολικά έξι μαθητές, οι τέσσερις με τη βοήθεια του DAVE, παρατήρησαν ότι αρκεί να αλλάξουν αμοιβαία θέσεις το πρώτο με το τελευταίο στοιχείο, αφού είναι τα μοναδικά που βρίσκονται σε λάθος θέσεις. Έτσι έδωσαν την εντολή του τμήματος κώδικα 8.6.

```
Αντιμετάθεσε A[ 1 ], A[ 6 ]
```

**Τμήμα Κώδικα 8.7. Αμοιβαία αλλαγή των αντιδιαμετρικών στοιχείων του πίνακα**

Η οπτικοποίηση της αμοιβαίας αλλαγής των δυο αυτών στοιχείων στο DAVE δίνεται στην οθόνη του Σχήματος 8.8.




**Σχήμα 8.8. Αμοιβαία αλλαγή των αντιδιαμετρικών στοιχείων του πίνακα στο DAVE**

### *8.4.3 Δραστηριότητα 3*

Η τρίτη δραστηριότητα συνίσταται σε δυο έργα. Στο πρώτο ζητείται από τους μαθητές να εκτελέσουν τον αλγόριθμο ταξινόμησης για έναν συγκεκριμένο πίνακα 5 στοιχείων, ενώ στο δεύτερο οι μαθητές καλούνται να σχεδιάσουν έναν αποδοτικό αλγόριθμο για τα συγκεκριμένα δεδομένα. Ακολουθεί η εκφώνηση της δραστηριότητας όπως δόθηκε στους μαθητές:

*Εισάγετε πίνακα με τα εξής πέντε στοιχεία: 1 4 3 2 5.*

***α)****Πόσα περάσματα θα χρειαστούν ώστε να ταξινομηθεί ο πίνακας σε αύξουσα σειρά;*

***β)****Μπορείτε να προτείνετε έναν τρόπο ταξινόμησης του πίνακα με πολύ λιγότερες αντιμεταθέσεις;*

*Να αιτιολογήσετε την απάντησή σας.*

Από τους 45 μαθητές, οι 25 απάντησαν ότι αρκούν 2 περάσματα χωρίς να χρειαστεί να χρησιμοποιήσουν το λογισμικό. Επίσης τρεις μαθητές δεν απάντησαν σε αυτό το ερώτημα. Πρόκειται για τους ίδιους τρεις μαθητές που δεν απάντησαν και στα δύο προηγούμενα ερωτήματα. Δεκατρείς μαθητές αξιοποίησαν το λογισμικό για να καταλήξουν στη σωστή απάντηση, ότι αρκούν δυο περάσματα, ώστε να ανέβουν στις θέσεις τους οι αριθμοί 2 και 3. Έντεκα από αυτούς είχαν δώσει αρχικά λανθασμένες απαντήσεις, όπως οι παρακάτω:

*«Απαιτούνται 3 περάσματα του αλγόριθμου ταξινόμησης»*

ενώ δύο δεν είχαν δώσει καμία απάντηση, χωρίς τη βοήθεια του λογισμικού. Οι δύο αυτοί μαθητές δεν είχαν απαντήσει στα περισσότερα από τα προηγούμενα ερωτήματα. Η δυνατότητα όμως που τους δόθηκε να δοκιμάσουν τον κώδικά τους στο λογισμικό οπτικοποίησης τους έδωσε κίνητρο να προσπαθήσουν να δώσουν μια απάντηση σε αυτό το ερώτημα.

**Πίνακας 8.6. Απαντήσεις μαθητών στη Δραστηριότητα 2α**

| **Απαντήσεις (N = 45)** | **Στο χαρτί** | **Με χρήση DAVE** |
|---|---|---|
| Δεν απάντησαν | 3 | 1 |
| Χρειάζονται δύο (2) περάσματα | 25 | 38 |
| Χρειάζονται τρία (3) περάσματα | 17 | 6 |

Το δεύτερο ερώτημα της δραστηριότητας ήταν αρκετά πιο απαιτητικό από το πρώτο, αφού αφορούσε τη βελτιστοποίηση του αρχικού τμήματος κώδικα, για αυτό σχεδόν οι μισοί μαθητές δεν προσπάθησαν να δώσουν μια απάντηση σε αυτό.

**Πίνακας 8.7. Απαντήσεις μαθητών στη Δραστηριότητα 2β**

| Απαντήσεις (N = 45) | Στο χαρτί | Με χρήση DAVE |
|---|---|---|
| Δεν απάντησαν | 24 | 21 |
| Προσπέλαση πέρα από τα όρια του πίνακα | 7 | 0 |
| Σύγκριση των στοιχείων στις ζυγές θέσεις | 1 | 15 |
| Λανθασμένη αντιμετάθεση | 4 | 0 |
| Ταξινόμηση όλου του πίνακα | 8 | 3 |
| Αντιμετάθεσε A[2], A[4] | 1 | 6 |

Ενδιαφέρον παρουσιάζει η προσέγγιση εννέα μαθητών, οι οποίοι παρατήρησαν ότι αρκεί να συγκριθούν τα στοιχεία που βρίσκονται στις ζυγές θέσεις και έδωσαν τον παρακάτω αλγόριθμο:

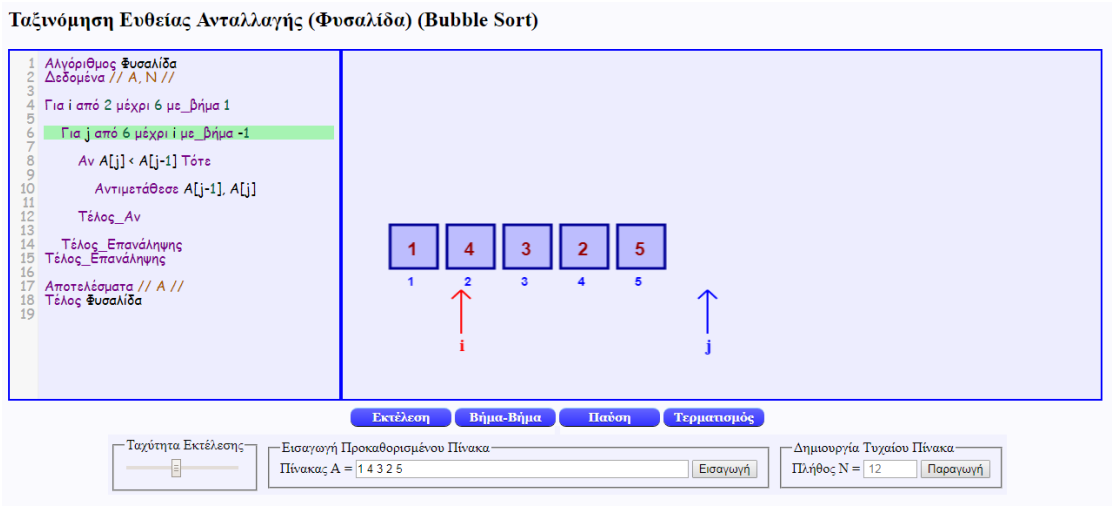
```
Για i από 2 μέχρι 6 με βήμα 2
    Για j από 6 μέχρι i με βήμα –2
        Αν A[ j ] < A[ j–2 ] Τότε Αντιμετάθεσε A[ j ], A[ j–2 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.8. Σύγκριση στοιχείων που βρίσκονται στις ζυγές θέσεις**

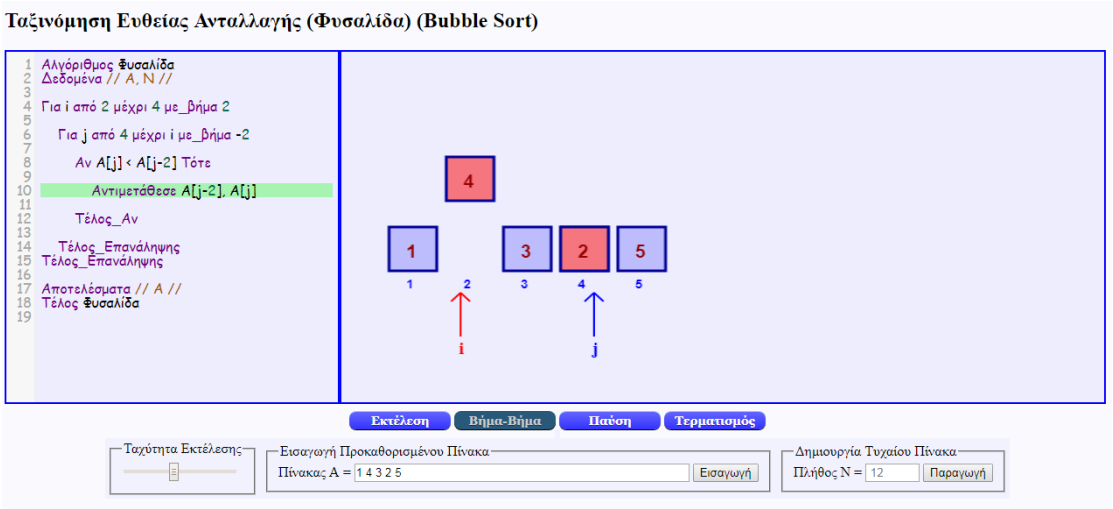
Κατά την εκτέλεση της οπτικοποίησης του αλγορίθμου οι μαθητές παρατήρησαν ότι ο δείκτης j έχει μετακινηθεί πέρα από το 5ο στοιχείο του πίνακα, όπως φαίνεται στο Σχήμα 8.9. Σε αυτό το σημείο αντιλήφθηκαν ότι έχουν χρησιμοποιήσει τον αριθμό 6 ως μέγεθος του πίνακα ενώ τα στοιχεία του είναι 5. Έτσι έδωσαν μια νέα διορθωμένη έκδοση της λύσης τους όπως φαίνεται στο Τμήμα Κώδικα 8.9. Εκτελώντας το Τμήμα Κώδικα 8.9 τα στοιχεία 4 και 2 που βρίσκονται στις θέσεις 2 και 4 αντίστοιχα αλλάζουν αμοιβαία θέση όπως φαίνεται στο Σχήμα 8.10. Εδώ φαίνεται η δυνατότητα που προσφέρει το λογισμικό να μπορεί να ορίσει κάποιος ποια στοιχεία θα συγκριθούν και θα αλλάξουν αμοιβαία θέση κατά την εκτέλεση του αλγορίθμου.

```
Για i από 2 μέχρι 4 με βήμα 2
    Για j από 4 μέχρι i με βήμα –2
        Αν A[ j ] < A[ j–2 ] Τότε Αντιμετάθεσε A[ j ], A[ j–2 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.9. Σύγκριση στοιχείων που βρίσκονται στις ζυγές θέσεις**

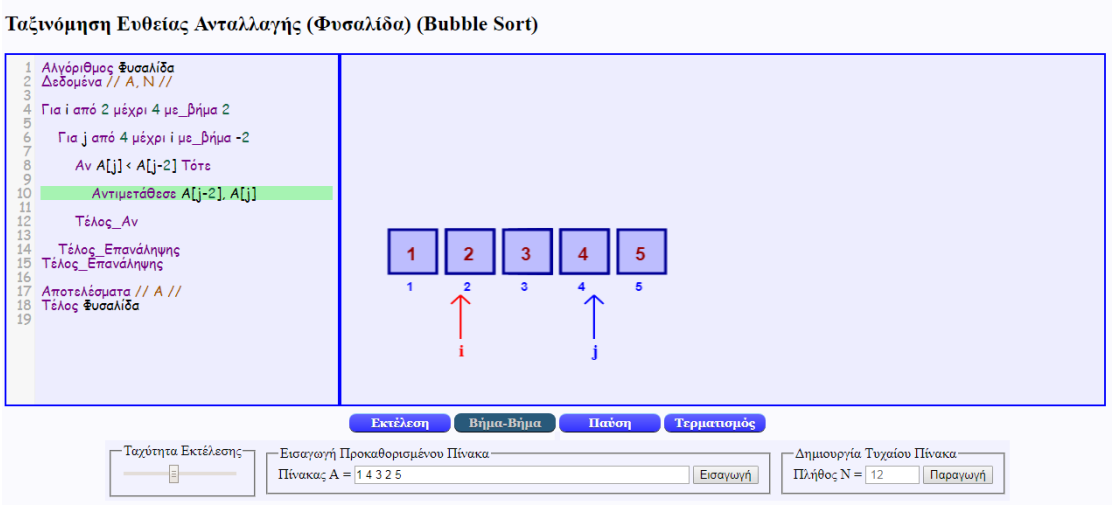



**Σχήμα 8.9. Ο δείκτης j έχει μετακινηθεί πέρα από τα όρια του πίνακα**



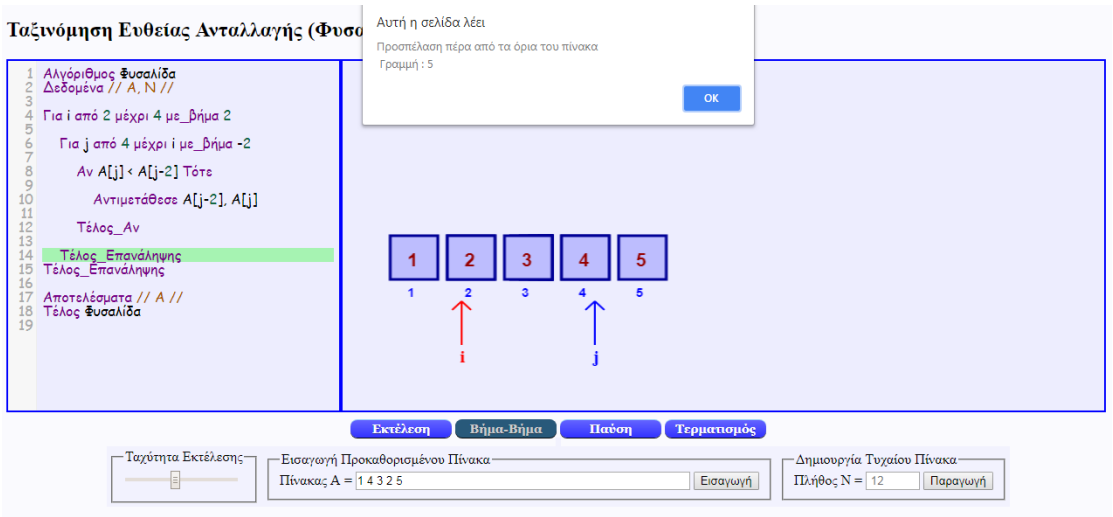

**Σχήμα 8.10. Τα στοιχεία 4 και 2 αλλάζουν αμοιβαία θέση**

Όταν οι μαθητές εκτέλεσαν το Τμήμα Κώδικα 8.9 εντόπισαν δυο βασικά χαρακτηριστικά του αλγορίθμου που σχεδίασαν τα οποία δεν είχαν αντιληφθεί πριν παρατηρήσουν την αντίστοιχη οπτικοποίηση. Το πρώτο ήταν ότι μετά τη μετακίνηση των δυο στοιχείων ο πίνακας είναι πλέον ταξινομημένος και δεν χρειάζεται άλλα περάσματα όπως φαίνεται στο Σχήμα 8.11.




**Σχήμα 8.11. Τα στοιχεία 4 και 2 αλλάζουν αμοιβαία θέση**

Το δεύτερο ήταν ότι για i=j=2 υπήρχε προσπέλαση πέρα από τα όρια του πίνακα στο στοιχείο A[j–2]=A[0], όπως φαίνεται στο Σχήμα 8.12.




**Σχήμα 8.12. Τα στοιχεία 4 και 2 αλλάζουν αμοιβαία θέση**

Το λάθος αυτό εντοπίστηκε εύκολα από τους μαθητές, καθώς το σύστημα εμφάνισε σχετικό διαγνωστικό μήνυμα. Στη συνέχεια επτά (7) μαθητές με συνεχείς αλλαγές διόρθωσαν τον αλγόριθμό τους καταλήγοντας στην παρακάτω λύση:

```
Για i από 2 μέχρι 4 με βήμα 2
    Για j από 4 μέχρι i+1 με βήμα –2
        Αν A[ j ] < A[ j–2 ] Τότε Αντιμετάθεσε A[ j ], A[ j–2 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.10. Τελική ορθή λύση**

Επίσης τρεις (3) μαθητές πρότειναν το Τμήμα Κώδικα 8.11 ως μια αποδοτική λύση στο πρόβλημα:

```
Για i από 2 μέχρι 2
    Για j από 3 μέχρι 3 με βήμα –2
        Αν A[ j ] < A[ j–2 ] Τότε Αντιμετάθεσε A[ j ], A[ j–2 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.11. Τελική αποδοτική λύση**

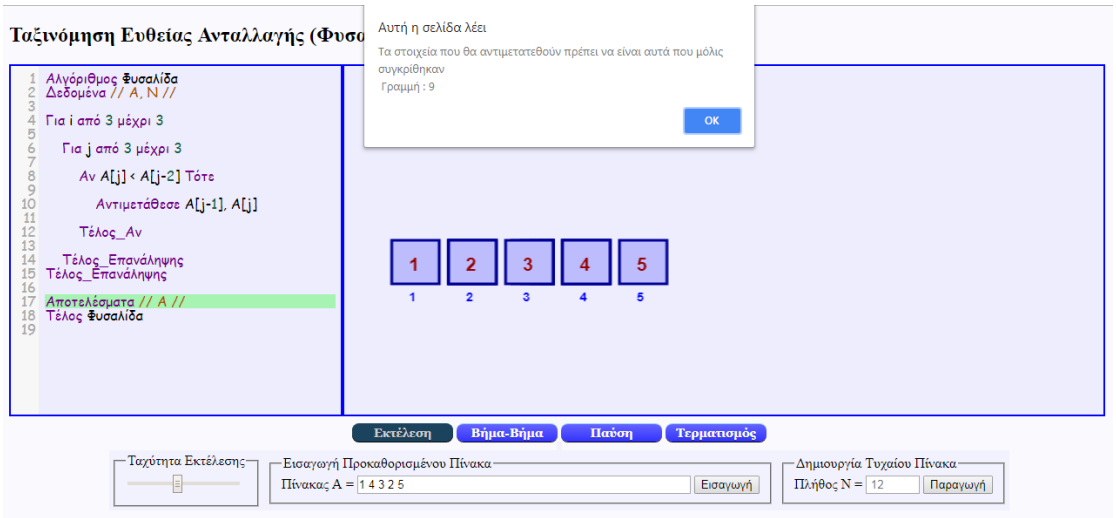
Τέλος, ένας μαθητής παρατήρησε ότι το παραπάνω τμήμα κώδικα μπορεί να γραφτεί ισοδύναμα χωρίς τη χρήση εντολής επανάληψης (Τμήμα Κώδικα 8.12) πριν ακόμα χρησιμοποιήσει το λογισμικό για να το επιβεβαιώσει. Άλλοι πέντε μαθητές κατέληξαν στην ίδια απάντηση ως βελτιστοποίηση του τμήματος κώδικα 8.11.

```
Αντιμετάθεσε A[ 4 ], A[ 2 ]
```

**Τμήμα Κώδικα 8.12. Τελική απλή λύση**

Εδώ οι μαθητές παρατήρησαν ότι αφού μόνο δυο στοιχεία (A[2], A[4]) είναι σε λάθος, αρκεί μια αμοιβαία αλλαγή μεταξύ τους.

Σε αυτή τη δραστηριότητα οι μαθητές προσάρμοσαν τον αλγόριθμο ευθείας ανταλλαγής έτσι ώστε να αξιοποιηθούν οι ιδιότητες των δεδομένων.




**Σχήμα 8.13. Διαγνωστικό μήνυμα όταν δεν αντιμετατίθενται τα σωστά στοιχεία.**

Στο Σχήμα 8.13 παρουσιάζεται ένα ακόμα λάθος στο οποίο υποπίπτουν οι μαθητές και το οποίο ανιχνεύει το λογισμικό. Στον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής αλλάζουν αμοιβαία θέση τα στοιχεία που μόλις πριν έχουν συγκριθεί. Μερικές φορές

οι μαθητές ξεχνούν να το ελέγξουν με αποτέλεσμα να αντιμετατίθενται άλλα στοιχεία από αυτά που συγκρίθηκαν κάτι που στον συγκεκριμένο αλγόριθμο δεν έχει νόημα. Το σύστημα ανιχνεύει το συγκεκριμένο πρόβλημα και ενημερώνει τους μαθητές με ένα διαγνωστικό μήνυμα όπως φαίνεται στο Σχήμα 8.13.

### *8.4.4 Δραστηριότητα 4*

*Εισάγετε τον πίνακα 7 στοιχείων 32 38 98 54 60 90 20*

*Να σχεδιάσετε τον αλγόριθμο ταξινόμησης της φυσαλίδας για την περίπτωση που η φυσαλίδα κατευθύνεται προς τα κάτω και όχι προς τα πάνω σε έναν πίνακα Α, Ν θέσεων. Δηλαδή ξεκινάει από τη θέση 1 και κινείται προς το τέλος του πίνακα όπως φαίνεται παρακάτω για έναν πίνακα 7 θέσεων.*

| | **1ο πέρασμα** | | | | | | **2ο πέρασμα** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 32 | 38 | 38 | 38 | 38 | 38 | 38 | 98 | 98 | 98 | 98 | 98 |
| 2 | 38 | 32 | 98 | 98 | 98 | 98 | 98 | 38 | 54 | 54 | 54 | 54 |
| 3 | 98 | 98 | 32 | 54 | 54 | 54 | 54 | 54 | 38 | 60 | 60 | 60 |
| 4 | 54 | 54 | 54 | 32 | 60 | 60 | 60 | 60 | 60 | 38 | 90 | 90 |
| 5 | 60 | 60 | 60 | 60 | 32 | 90 | 90 | 90 | 90 | 90 | 38 | 38 |
| 6 | 90 | 90 | 90 | 90 | 90 | 32 | 32 | 32 | 32 | 32 | 32 | 32 |
| 7 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |

*Σημειώστε τον τρόπο σκέψης σας και τις δοκιμές ή τα λάθη που έγιναν μέχρι να καταλήξετε στον ζητούμενο αλγόριθμο.*

Η δραστηριότητα αυτή θεωρείται αυξημένης δυσκολίας, καθώς απαιτεί βαθιά κατανόηση του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής. Θα πρέπει να σημειωθεί ότι μια σχετική αλλά πιο απλή δραστηριότητα ανάγνωσης κώδικα είχε τεθεί στην έρευνα ανάδειξης και καταγραφής των δυσκολιών των μαθητών στον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής (Βραχνός & Τζιμογιάννης, 2014α), η οποία παρουσιάζεται αναλυτικά στο κεφάλαιο 6 της παρούσας διατριβής. Η έρευνα έδειξε ότι μόνο το 20% των μαθητών και το 36% των φοιτητών έδωσε σωστή απάντηση.

Στην παρούσα έρευνα το πρόβλημα είναι ακόμα πιο δύσκολο γιατί ζητά από τους μαθητές να επανασχεδιάσουν ουσιαστικά μια παραλλαγή του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής. Είναι χαρακτηριστικό ότι μόνο ένας από τους 45 μαθητές έδωσε σωστή λύση χωρίς τη βοήθεια του λογισμικού οπτικοποίησης.

Είκοσι (20) μαθητές δεν ασχολήθηκαν καθόλου με το πρόβλημα αυτό, με την αιτιολόγηση ότι το θεώρησαν πολύ δύσκολο. Οι μαθητές που ασχολήθηκαν με το θέμα τροποποίησαν από την αρχή την εσωτερική επανάληψη της ταξινόμησης ευθείας ανταλλαγής, έτσι ώστε τα μικρότερα στοιχεία να κατευθύνονται προς τα κάτω, ως αντίστροφη φυσαλίδα (sinking sort). Από αυτούς, 16 έφτασαν στη σωστή απάντηση μετά από διαδοχικούς πειραματισμούς με το περιβάλλον οπτικοποίησης. Αρκετοί μαθητές σημείωσαν στο φύλλο αξιολόγησης, που τους δόθηκε μετά το πέρας της έρευνας, ότι τους βοήθησε πολύ το γεγονός ότι, κατά την εκτέλεση του αλγορίθμου, το σύστημα εντοπίζει την προσπέλαση πέρα από τα όρια του πίνακα.

**Πίνακας 8.8. Απαντήσεις μαθητών στη Δραστηριότητα 2β**

| Απαντήσεις (N = 45) | Στο χαρτί | Με χρήση DAVE |
|---|---|---|
| Δεν απάντησαν | 31 | 21 |
| Προσπέλαση πέρα από τα όρια του πίνακα | 7 | 0 |
| ημιτελής λύση | 1 | 15 |
| Σωστός αλγόριθμος | 1 | 6 |

Τέσσερις μαθητές έφτασαν σε μια ημιτελή λύση από την οποία όμως φαινόταν ότι είχαν κατανοήσει τη λογική του αλγορίθμου. Τα λάθη τους εντοπίζονταν κυρίως σε λεπτομέρειες παρά στη βασική φιλοσοφία του αλγορίθμου που ήταν σωστή. Για παράδειγμα, στη συνέχεια παρουσιάζουμε έναν χαρακτηριστικό τρόπο σκέψης, όπως τον αποτύπωσε μια μαθήτρια στο φύλλο εργασίας της. Στην αρχή σχεδίασε τον παρακάτω αλγόριθμο.

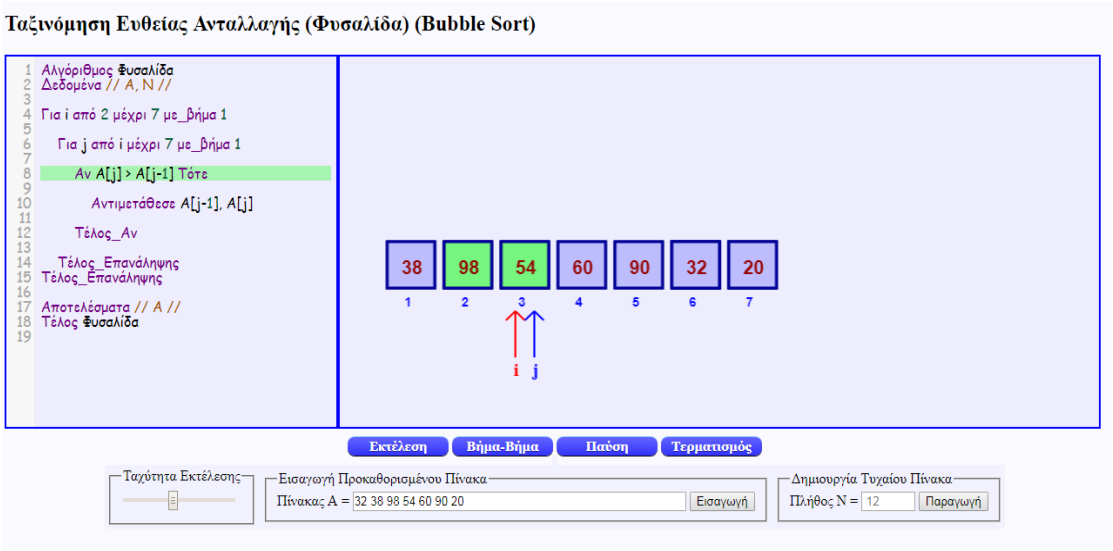
```
Για i από 2 μέχρι 7
    Για j από i μέχρι 7
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.13. Προσπέλαση πέρα από τα όρια του πίνακα**

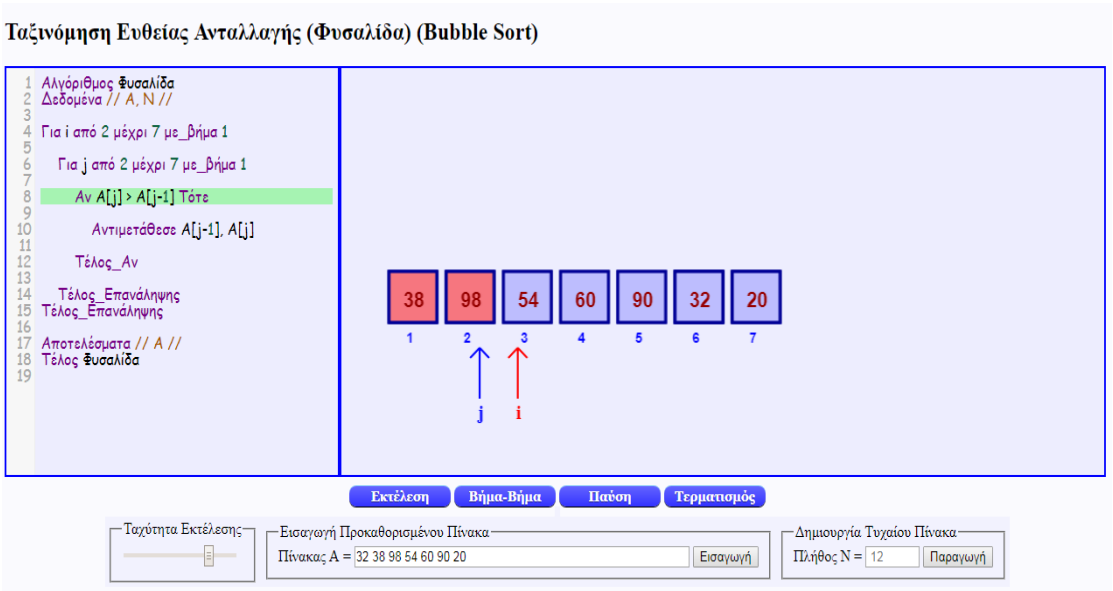
Παρακολουθώντας την οπτικοποίηση του τμήματος κώδικα, η μαθήτρια παρατήρησε ότι ο αλγόριθμος αυτός δεν εκτελεί ταξινόμηση, διότι ενώ το i αυξάνεται, δηλαδή ο αντίστοιχος δείκτης μετακινείται προς τα δεξιά το μέρος του πίνακα που απομένει πριν τον δείκτη δεν είναι ταξινομημένο όπως θα έπρεπε. Στο Σχήμα 8.14 το

στοιχείο 98 δεν μπορεί να ανέβει στην πρώτη θέση αφού ο δείκτης i έχει προχωρήσει παρακάτω.

Στη συνέχεια η μαθήτρια άλλαξε την αρχική τιμή του j σε 2 έτσι ώστε όχι μόνο το 32 να βρεθεί στην προτελευταία θέση αλλά και το 98 να ανέβει στην πρώτη θέση στο επόμενο πέρασμα αφού είναι το μεγαλύτερο. Όπως φαίνεται στο Σχήμα 8.15 στο δεύτερο πέρασμα (i=3) συγκρίνονται πάλι τα δυο πρώτα στοιχεία αφού ο δείκτης j είναι ανεξάρτητος του i και ξεκινάει πάλι από 2.




**Σχήμα 8.14. Το στοιχείο 98 έπρεπε να βρίσκεται στην πρώτη θέση**



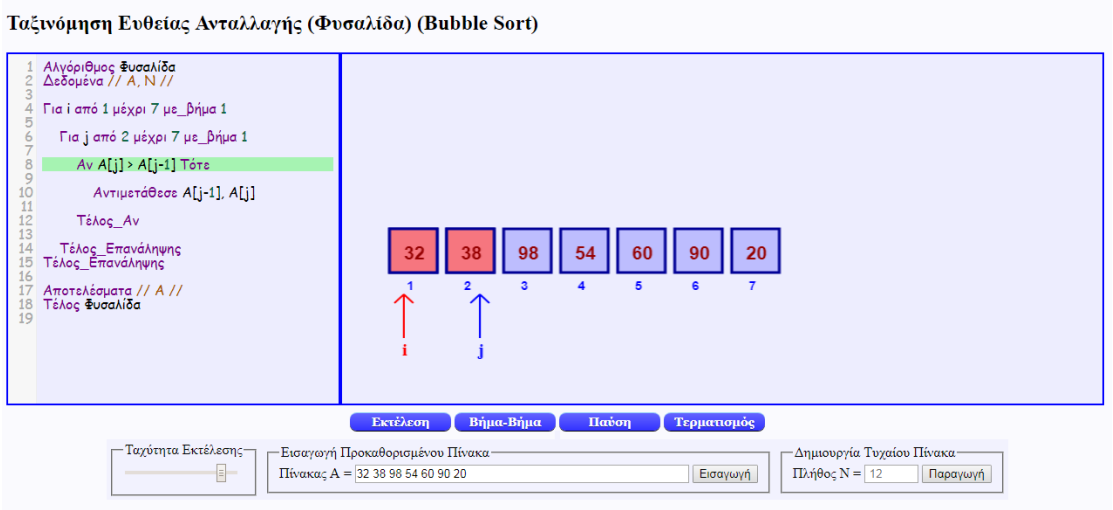

**Σχήμα 8.15. Στο δεύτερο πέρασμα συγκρίνονται πάλι τα δυο πρώτα στοιχεία**

Η μαθήτρια συνέχισε να πειραματίζεται και άλλαξε την αρχική τιμή του i από 2 σε 1, αιτιολογώντας την επιλογή της αυτή ως εξής:

*«Δεν φαίνεται σωστό όταν ξεκινάει η ταξινόμηση με τα δυο πρώτα στοιχεία, ο δείκτης i να δείχνει στο δεύτερο στοιχείο.»*

Η προσέγγιση που ακολούθησε η μαθήτρια δείχνει ότι οι μαθητές συγχέουν το ρόλο του δείκτη i της εξωτερικής επανάληψης με το ρόλο του δείκτη j. Ο δείκτης i καθορίζει πόσα περάσματα θα γίνουν (πόσες φυσαλίδες θα ανέβουν) ενώ ο δείκτης j είναι που αναφέρεται στα στοιχεία του πίνακα που συγκρίνονται σε κάθε βήμα. Η σύγχυση των ρόλων των δεικτών των εντολών επανάληψης, αποτελεί μια από τις βασικές παρανοήσεις που έχουν οι μαθητές στον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής.

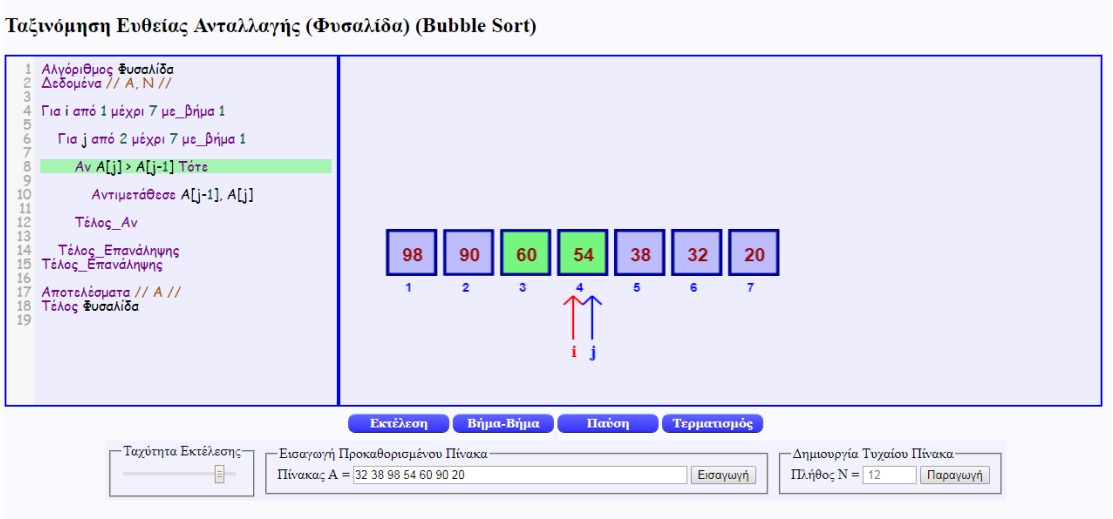
Η μαθήτρια στη συνέχεια παρατήρησε ότι τα στοιχεία του πίνακα έχουν ταξινομηθεί μετά το 4ο πέρασμα. Στη συνέχεια δεν γίνεται καμία αντιμετάθεση, με αποτέλεσμα να είναι περιττές όλες οι συγκρίσεις που γίνονται όπως φαίνεται στο Σχήμα 8.17. Έτσι, έδωσε μια ακόμη εκδοχή της λύσης της με το Τμήμα Κώδικα 8.14, όπου εκτελούνται ακριβώς τέσσερα περάσματα του αλγορίθμου, διότι η μαθήτρια διέκρινε ότι τόσα χρειάζονται για τα συγκεκριμένα δεδομένα




**Σχήμα 8.16. Ο δείκτης i ξεκινάει από το πρώτο στοιχείο**

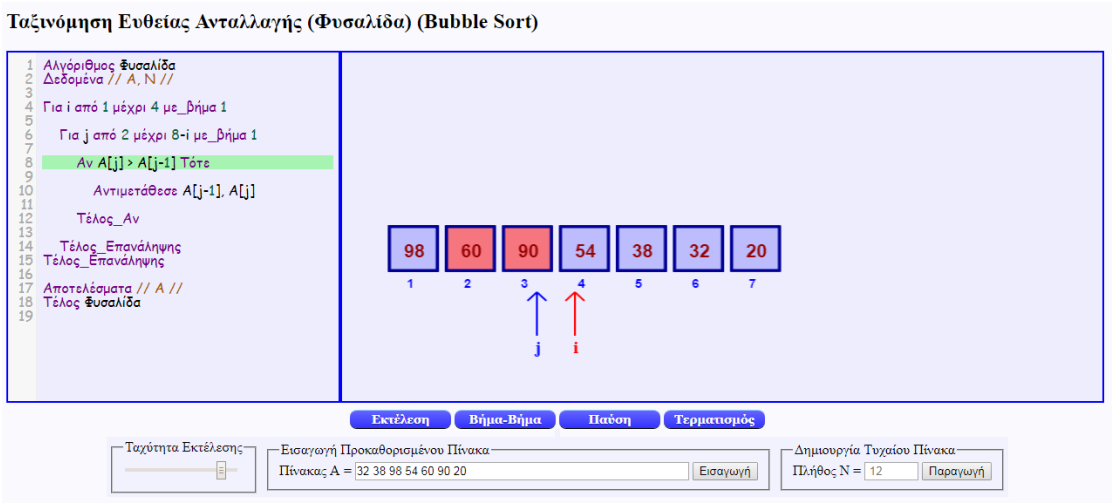
```
Για i από 1 μέχρι 4
    Για j από 2 μέχρι 7
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.14. Αποδοτική Λύση με τέσσερα μόνο περάσματα**

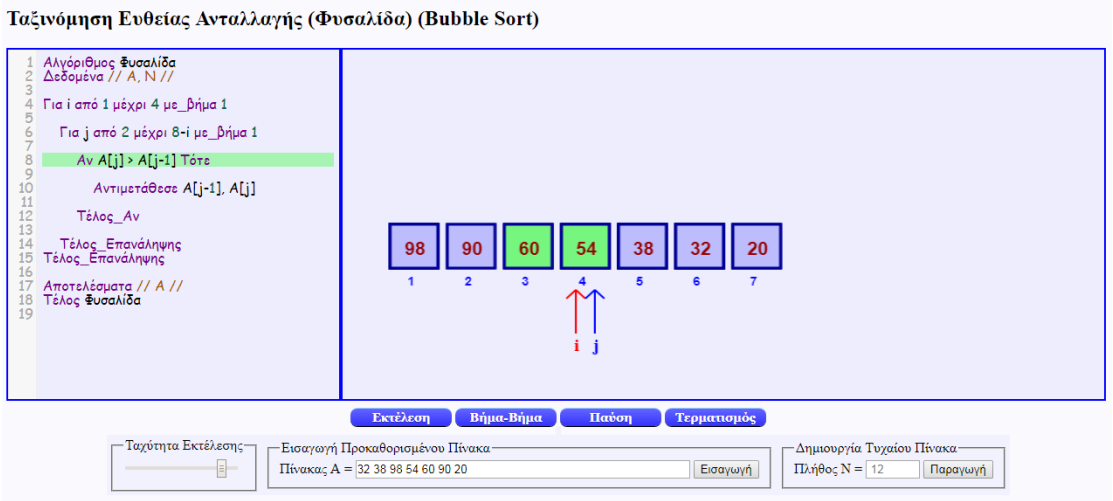



**Σχήμα 8.17. Τα στοιχεία του πίνακα είναι ήδη ταξινομημένα όταν i=4, j=4.**

Το Τμήμα Κώδικα 8.14 δεν ήταν το τελικό, αφού η μαθήτρια έκανε μια τελευταία αλλαγή στον κώδικα. Παρατήρησε μετά από κάποιο σημείο τα στοιχεία του πίνακα είναι ήδη ταξινομημένα από τα προηγούμενα περάσματα. Για παράδειγμα στην οθόνη του DAVE που φαίνεται στο Σχήμα 8.17, τα τρία τελευταία στοιχεία είναι τα μικρότερα και έχουν τοποθετηθεί στις ήδη στις θέσεις τους, άρα δεν χρειάζεται οι συγκρίσεις να φτάσουν πέρα από το στοιχείο στη θέση 5. Αυτό φαίνεται και στο Σχήμα 8.18, όπου δίνεται η οθόνη της τελικής λύσης της μαθήτριας. Εκεί φαίνεται ότι τα στοιχεία στις θέσεις 2 και 3 είναι τα τελευταία που θα αλλάξουν αμοιβαία θέση αφού όλα τα άλλα στοιχεία έχουν τοποθετηθεί στις τελικές θέσεις τους.




**Σχήμα 8.18. Τα δυο τελευταία στοιχεία που αλλάζουν θέση στην τελική λύση**




**Σχήμα 8.19. Οι θέσεις των δεικτών i, j όταν ο πίνακας έχει ταξινομηθεί**

```
Για i από 1 μέχρι 4
    Για j από 2 μέχρι 8–i
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.15. Η τελική αποδοτική λύση της μαθήτριας**

Στη συνέχεια όπως φαίνεται στην οθόνη του σχήματος 8.19 όλα τα υπόλοιπα στοιχεία έχουν πράσινη ένδειξη που σημαίνει ότι είναι στη σωστή σειρά. Όλα αυτά τα παρατήρησε η μαθήτρια μετά από εκτεταμένους πειραματισμούς με το λογισμικό ώστε να καταλήξει στο Τμήμα Κώδικα 8.15.

Η μαθήτρια όταν ρωτήθηκε για τον τρόπο εργασίας της, απάντησε ότι ξεκίνησε αρχικά με τον αλγόριθμο της ευθείας ανταλλαγής που γνώριζε, τον οποίο προσπάθησε να τροποποιήσει. Ανάλογη προσέγγιση ακολούθησαν και οι περισσότεροι μαθητές, οι οποίοι προσπάθησαν να χτίσουν πάνω σε προϋπάρχουσα γνώση, δηλαδή στον αλγόριθμο ευθείας ανταλλαγής. Στη συνέχεια όμως κατέληξαν να σχεδιάσουν τον αλγόριθμο από την αρχή. Όπως έγραψαν στο φύλλο εργασίας, τους βοήθησε πολύ η δυνατότητα εντοπισμού λογικών λαθών που παρείχε το λογισμικό είτε μέσω παρατήρησης της οπτικοποίησης είτε μέσω διαγνωστικών μηνυμάτων, ώστε να καταλήξουν στην τελική τους απάντηση.

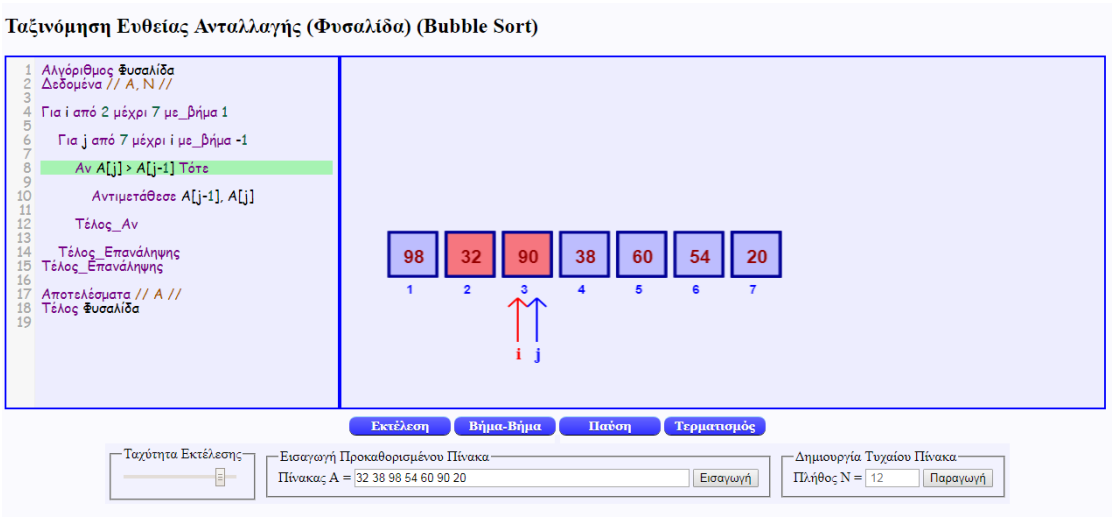
Στη συνέχεια θα αναλύσουμε τον τρόπο σκέψης ενός δεύτερου μαθητή με βάση το φύλλο εργασίας του αλλά και τη συζήτηση που κάναμε μαζί του. Ο μαθητής ξεκίνησε

με το Τμήμα Κώδικα 8.16 το οποίο εκτελεί φθίνουσα ταξινόμηση των στοιχείων του πίνακα. Κατά την εκτέλεση του αλγορίθμου από το λογισμικό ο μαθητής παρατήρησε ότι τα μεγαλύτερα στοιχεία κινούνται προς τις πρώτες θέσεις του πίνακα και όχι τα μικρότερα προς τις τελευταίες όπως φαίνεται στην οθόνη του λογισμικού στο Σχήμα 8.20.

Παρόλο που ο αλγόριθμος εκτελούσε ταξινόμηση του πίνακα δεν επιτελούσε τη ζητούμενη λειτουργία. Αυτό οδήγησε τον μαθητή να διορθώσει τον αλγόριθμο έτσι ώστε τα μικρότερα στοιχεία να μετακινούνται προς το τέλος του πίνακα (Τμήμα Κώδικα 8.16).

```
Για i από 2 μέχρι 7
    Για j από 7 μέχρι i με βήμα –1
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

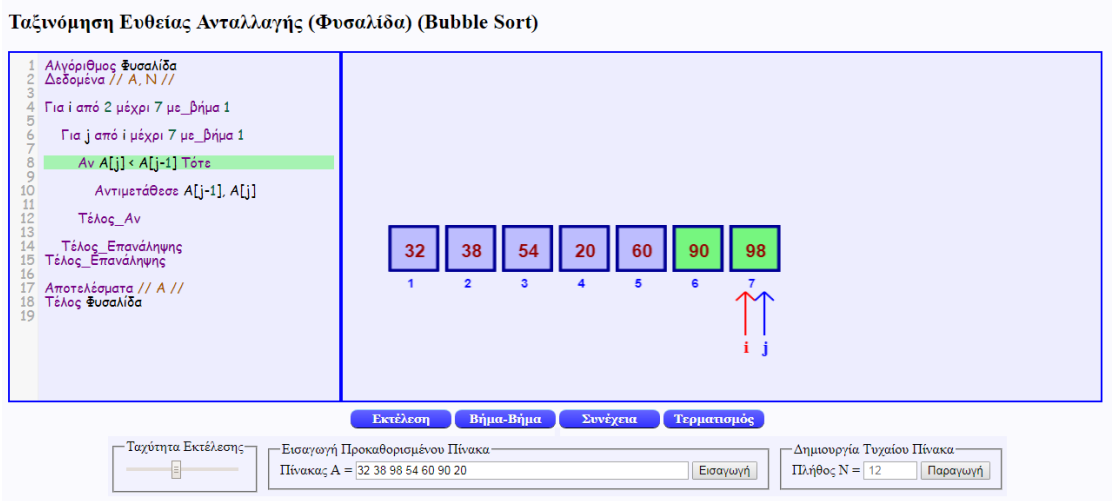
**Τμήμα Κώδικα 8.17. Φθίνουσα ταξινόμηση των στοιχείων: Αρχικός αλγόριθμος**




**Σχήμα 8.20. Τα μεγαλύτερα στοιχεία κινούνται στις πρώτες θέσεις του πίνακα**

```
Για i από 2 μέχρι 7
    Για j από i μέχρι 7
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.18. Τα μικρότερα στοιχεία κινούνται προς τις τελευταίες θέσεις**




**Σχήμα 8.21. Μετά το τέλος του αλγορίθμου τα στοιχεία δεν είναι ταξινομημένα**

```
Για i από 2 μέχρι 7
    Για j από 2 μέχρι 7
        Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
    Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.19. Ορθή λύση στο πρόβλημα**

Εκτελώντας πάλι όμως τον αλγόριθμο ο μαθητής παρατήρησε ότι στο τέλος τα στοιχεία δεν είναι ταξινομημένα ( 32 98 60 90 54 38 20 ), όπως φαίνεται στην οθόνη του Σχήματος 8.21. Αυτό τον οδήγησε στην τροποποίηση των ακραίων τιμών της εσωτερικής επανάληψης όπως φαίνεται στο Τμήμα Κώδικα 8.18.

Κατά την εκτέλεση της οπτικοποίησης του νέου αλγορίθμου ο μαθητής παρατήρησε ότι το αμέσως μικρότερο στοιχείο, που κινείται προς τις τελευταίες θέσεις του πίνακα, σταματάει μόλις φτάσει στο ταξινομημένο τμήμα του. Όμως, οι συγκρίσεις συνεχίζονται και για τα τελευταία στοιχεία που είναι ήδη στις σωστές θέσεις. Ουσιαστικά ο μαθητής σχεδίασε σιγά σιγά από την αρχή τον αλγόριθμο ταξινόμησης με μικρές αλλαγές σταδιακά.

Στην αρχή έθεσε ως άνω όριο στην εσωτερική επανάληψη την τιμή 8–i την οποία διόρθωσε στη συνέχεια σε 9–i, αφού μελέτησε την οπτικοποίηση σε αυτή την περίπτωση. Μετά από εκτεταμένο πειραματισμό με το λογισμικό ο μαθητής κατάλαβε ότι η εσωτερική επανάληψη καθορίζει πόσο θα συνεχιστούν οι συγκρίσεις σε κάθε πέρασμα. Επίσης παρατηρώντας ένα προς ένα τα περάσματα του αρχικού αλγορίθμου

σε συνδυασμό με τη γραφική απεικόνιση των δεικτών i και j και τον χρωματισμό των στοιχείων που συγκρίνονται σε κάθε βήμα, διέκρινε κάθε φορά οι συγκρίσεις είναι όλο και λιγότερες, αφού το ταξινομημένο τμήμα του πίνακα μεγαλώνει μετά από κάθε πέρασμα.

Στη συνέχεια με διαδοχικές δοκιμές των τιμών της τελικής τιμής της εσωτερικής επανάληψης έφτασε στην τελική λύση που δίνεται στο Τμήμα Κώδικα 8.19.

```
Για i από 2 μέχρι 7
   Για j από 2 μέχρι 9 – i
      Αν A[ j ] > A[ j–1 ] Τότε Αντιμετάθεσε A[ j ], A[ j–1 ]
   Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

**Τμήμα Κώδικα 8.20. Τελική αποδοτική λύση στο πρόβλημα**

Η πορεία που ακολούθησε ο δεύτερος μαθητής μοιάζει σε κάποια σημεία με την πορεία που ακολούθησε η μαθήτρια. Ωστόσο η διαφορά της κάθε προσέγγισης έγκειται στο γεγονός ότι η μαθήτρια θεώρησε ότι ο μετρητής i της εξωτερικής επανάληψης πρέπει να ξεκινάει οπωσδήποτε από το 1, ενώ ο μαθητής όχι. Η διαφορά αυτή έχει να κάνει με τον ρόλο που αποδίδουν στη μεταβλητή μετρητή i της εξωτερικής επανάληψης οι δυο μαθητές. Η μαθήτρια τη συνδέει με τις θέσεις των στοιχείων του πίνακα ενώ ο μαθητής έχει κατανοήσει ότι αυτή η μεταβλητή ορίζει τα περάσματα που χρειάζεται να γίνουν μέχρι ο πίνακας να ταξινομηθεί.

## *8.5 Παρατηρήσεις - Σχόλια μαθητών*

Το περιβάλλον οπτικοποίησης DAVE ενεργοποίησε σε μεγάλο βαθμό τους μαθητές, ενίσχυσε τη γνωστική εμπλοκή τους για την επίλυση αλγοριθμικών προβλημάτων μέσα από τον πειραματισμό και τη διερεύνηση εναλλακτικών εκδοχών του αλγορίθμου, την αναθεώρηση και, τελικά, τη βελτίωση των δικών τους αλγορίθμων. Οι περισσότεροι μαθητές ασχολήθηκαν με όλες δραστηριότητες του φύλλου εργασίας, ακόμη και με τα πιο απαιτητικά ερωτήματα. Είναι χαρακτηριστικό ότι στην έρευνα που παρουσιάστηκε στο 6ο κεφάλαιο της διατριβής και η οποία είχε ως σκοπό τη μελέτη της επίλυσης αλγοριθμικών προβλημάτων από τους μαθητές με χαρτί και μολύβι, ελάχιστοι ασχολήθηκαν με ερωτήματα αντίστοιχης δυσκολίας (Βραχνός & Τζιμογιάννης 2014α). Η διαθεσιμότητα του λογισμικού έδωσε τη δυνατότητα σε πολλούς μαθητές οι οποίοι σε άλλη περίπτωση θα τα παρατούσαν, να

προσπαθήσουν να σχεδιάσουν μια λύση έστω και αν δεν κατέληξαν στο σωστό αποτέλεσμα. Τους δόθηκε όμως η ευκαιρία μέσω της χρήσης του λογισμικού να κάνουν συλλογισμούς να σχεδιάσουν νέους τρόπους επίλυσης, να τους δοκιμάσουν και να τους διορθώσουν, κάτι που χωρίς τη χρήση του λογισμικού οπτικοποίησης θα ήταν ανέφικτο. Αυτό οφείλεται στην ανάδραση που δίνει το λογισμικό μέσω της οπτικοποίησης του αλγορίθμου για διάφορες κατηγορίες δεδομένων.

Στους μαθητές δόθηκε επίσης ειδικό φύλλο αξιολόγησης του περιβάλλοντος DAVE. Οι μαθητές, στην συντριπτική τους πλειονότητα έγραψαν ότι η ενασχόληση με το λογισμικό τους βοήθησε σημαντικά στο να κατανοήσουν τη λειτουργία του αλγορίθμου της ευθείας ανταλλαγής, και θεωρούν απαραίτητη τη χρήση του για το συγκεκριμένο μάθημα. Ένα από τα βασικά σχόλια των μαθητών ήταν ότι το λογισμικό τους επέτρεψε να πειραματιστούν με τον αλγόριθμο που είχαν σχεδιάσει και με συνεχείς αλλαγές, να καταλήξουν στη σωστή απάντηση. Αρκετοί σημείωσαν ότι τους βοήθησε πολύ στην κατανόηση του αλγορίθμου επειδή έβλεπαν την εκτελούμενη εντολή και ακριβώς δίπλα το αποτέλεσμα της οπτικοποίησής της.

Ακολουθούν μερικές παρατηρήσεις μαθητών όπως διατυπώθηκαν στο φύλλο αξιολόγησης:

«*Με βοήθησε να καταλάβω πως δουλεύει ο αλγόριθμος ταξινόμησης.*»

«*Η οπτικοποίηση των αλγορίθμων με βοήθησε πολύ να καταλάβω πως δουλεύουν.*»

«*Με βοήθησε πολύ ότι όταν άλλαζα τον αλγόριθμο έβλεπα αμέσως την οπτικοποίηση. Αυτό με βοήθησε στη διόρθωση των λαθών που έκανα.*»

«*Το ότι μπορούσα να αλλάζω κάθε φορά τα δεδομένα εισόδου του αλγόριθμου με βοήθησε πολύ στο να καταλάβω τη λειτουργία του.*»

«*Η εικόνα των μετρητών i ,j των επαναλήψεων με βοήθησε πολύ.*»

«*Θα βοηθούσε πολύ αν μας επέτρεπαν να χρησιμοποιήσουμε το λογισμικό στις εξετάσεις για να ελέγχουμε τα προγράμματά μας.*»

«*Πιστεύω ότι η δυνατότητα να χρησιμοποιούμε το πρόγραμμα και από το σπίτι θα βοηθήσει πολύ.*»

«*Μου άρεσε που μπορούσα να δω τον αλγόριθμο από το κινητό μου. Δυστυχώς δεν μου επιτρέπεται να το χρησιμοποιώ στο μάθημα.*»

**Πίνακας 8.9. Σχόλια-Παρατηρήσεις των μαθητών για το DAVE**

| Απαντήσεις (N = 45) | Θετική γνώμη (%) | Αρνητική γνώμη (%) |
|---|---|---|
| Ήταν εύκολο για μένα να εξοικειωθώ με το λογισμικό οπτικοποίησης | 89 | 11 |
| Ήταν εύκολο για μένα να χρησιμοποιήσω το λογισμικό οπτικοποίησης | 89 | 11 |
| Η οπτικοποίηση των μετρητών i,j των εντολών επανάληψης με βοήθησε | 78 | 22 |
| Η οπτικοποίηση με βοήθησε να καταλάβω τη λειτουργία του αλγορίθμου ταξινόμησης | 96 | 4 |
| Το λογισμικό οπτικοποίησης βοηθά πολύ στην επίλυση αλγοριθμικών προβλημάτων | 98 | 2 |

Αρκετοί μαθητές θεώρησαν πολύ σημαντικό ότι μπορούν να έχουν πρόσβαση στο λογισμικό από το σπίτι τους μέσω της αντίστοιχης ιστοσελίδας, ή ακόμα και από το κινητό τους τηλέφωνο.

Ένα άλλο σημείο που αξιολογήθηκε θετικά από τους μαθητές ήταν η κίνηση των μεταβλητών i, j και η γραφική απεικόνισή τους ως δείκτες στα αντίστοιχα στοιχεία του πίνακα. Η μετακίνηση του δείκτη στο επόμενο στοιχείο ως οπτικοποίηση της αύξησής του διευκολύνει τους μαθητές να αντιμετωπίσουν μια γνωστή δυσκολία που έχει εντοπιστεί στη βιβλιογραφία (Danielsiek 2012) σχετικά με τις εντολές επανάληψης στις οποίες δεν φαίνεται άμεσα η αύξηση του μετρητή. Για παράδειγμα στην εντολή Για i από 1 μέχρι 10 όπου το βήμα παραλείπεται δεν φαίνεται η αύξηση του δείκτη i.

Από τις σημειώσεις του ερευνητή προέκυψε ότι ελάχιστοι μαθητές έκαναν χρήση της βήμα προς βήμα εκτέλεσης με στόχο να επικεντρωθούν σε συγκεκριμένα σημεία του αλγορίθμου. Συνήθως, οι μαθητές επέλεγαν να εκτελούν τον αλγόριθμο πολλές φορές και να μειώνουν την ταχύτητα της εκτέλεσης στα σημεία ενδιαφέροντος. Επίσης κάποιοι μαθητές θα ήθελαν να υπάρχει δυνατότητα αντίστροφης (προς τα πίσω) εκτέλεσης, ώστε να εστιάζουν σε δύσκολα σημεία του αλγορίθμου χωρίς να εκτελούν κάθε φορά τον αλγόριθμο από την αρχή. Η αντίστροφη εκτέλεση του αλγορίθμου είναι μια δυνατότητα που αναμένεται να προστεθεί σε μελλοντική έκδοση του DAVE.

## *8.6 Συμπεράσματα*

Τα αποτελέσματα της εργασίας αυτής επιβεβαιώνουν τα ευρήματα προηγούμενων ερευνών για τη συμβολή των περιβαλλόντων δυναμικής οπτικοποίησης αλγορίθμων

στην ανάπτυξη προγραμματιστικών ικανοτήτων (Cetin & Andrews-Larson, 2016· Hundhausen, Douglas & Stasko, 2002· Shi, Min & Zhang, 2017· Sorva et al., 2013· Urquiza–Fuentes et al., 2009· Velazquez-Iturbide, Hernan-Losada, & Paredes-Velasco, 2017). Η χρήση του λογισμικού DAVE ενίσχυσε την προσπάθεια των μαθητών του δείγματος για επίλυση αλγοριθμικών προβλημάτων με πίνακες, προωθώντας την ενεργό συμμετοχή και τον πειραματισμό τους με τις οπτικοποιήσεις αλγορίθμων. Οι μαθητές αξιοποίησαν σχεδόν όλες τις δυνατότητες του λογισμικού και, κυρίως, την τροποποίηση του κώδικα του αλγορίθμου και την αυξομείωση της ταχύτητας της εκτέλεσης της οπτικοποίησης. Τα δυναμικά χαρακτηριστικά του λογισμικού και οι δυνατότητες εντοπισμού λογικών λαθών που υποστηρίζει το λογισμικό, όπως η προσπέλαση πέρα από τα όρια του πίνακα, συνέβαλαν στην ανάδειξη λογικών σφαλμάτων και στην οικοδόμηση επαρκών αναπαραστάσεων των μαθητών για τις έννοιες του πίνακα, του δείκτη, της αντιμετάθεσης και της σύγκρισης στοιχείων.

Ακολουθώντας στρατηγικές διερευνητικής μάθησης, και αξιοποιώντας την προσομοίωση του αλγορίθμου μέσω του λογισμικού DAVE, οι μαθητές είχαν την ευκαιρία να δοκιμάσουν την δική τους εκδοχή του αλγορίθμου επίλυσης του προβλήματος και να πειραματιστούν με αυτήν, εντοπίζοντας τυχόν λογικά λάθη μέσα από την οπτικοποίηση. Η πειραματική καταγραφή των διαδοχικών βημάτων προσέγγισης ανέδειξε δυσκολίες, παρανοήσεις και μοντέλα σκέψης των μαθητών που έχουν μεγάλη διδακτική αξία.

Επίσης μέσα από τον πειραματισμό με την οπτικοποίηση του αλγορίθμου οι μαθητές είχαν την ευκαιρία να μελετήσουν την απόδοση του αλγορίθμου για τα συγκεκριμένα δεδομένα, αφού μπορούσαν να διαπιστώσουν αν τα στοιχεία του πίνακα είχαν ταξινομηθεί πριν το τέλος του αλγορίθμου. Για να αποφύγουν περιττούς υπολογισμούς οι μαθητές είχαν την ευκαιρία να προτείνουν και να δοκιμάσουν βελτιωμένες εκδοχές του αλγορίθμου ως προς την απόδοση.

Τα αποτελέσματα έδειξαν ότι οι δυνατότητες πειραματισμού με την οπτικοποίηση της εκτέλεσης του αλγορίθμου, ενθάρρυναν τη συμμετοχή και την ενεργοποίηση των μαθητών κατά την επίλυση προβλημάτων ταξινόμησης πινάκων αυξημένης δυσκολίας. Έδωσαν δε, σημαντικές πληροφορίες για τη συμβολή του λογισμικού στην εξέλιξη της αλγοριθμικής σκέψης των μαθητών με στόχο την οικοδόμηση αλγόριθμων ταξινόμησης.

# 9

# *Συζήτηση - Συμπεράσματα*

## *9.1 Κύρια ευρήματα της διατριβής και ερμηνεία τους*

Στο πλαίσιο της διατριβής μελετήθηκαν θέματα που εντάσσονται στις περιοχές της Διδακτικής του Προγραμματισμού. Ο βασικός στόχος της διατριβής ήταν ο σχεδιασμός και η αξιολόγηση ενός λογισμικού οπτικοποίησης αλγορίθμων για την αντιμετώπιση των δυσκολιών που συναντούν οι μαθητές στις έννοιες του πίνακα και των αλγορίθμων ταξινόμησης. Αρχικά πραγματοποιήθηκαν δύο έρευνες σε μαθητές λυκείων και σε πρωτοετείς φοιτητές πληροφορικής για τη μελέτη των παρανοήσεων και των δυσκολιών σε αυτές τις έννοιες.

Η πρώτη έρευνα έγινε σε ένα δείγμα 102 μαθητών λυκείου και είχε στόχο τη μελέτη των αναπαραστάσεων των μαθητών για την έννοια του πίνακα. Η ανάλυση των απαντήσεων των μαθητών έγινε με βάση την ταξινομία SOLO. Τα αποτελέσματα έδειξαν ότι οι μαθητές έχουν ελλιπείς αναπαραστάσεις για την έννοια του πίνακα και τη χρησιμότητά του στην επίλυση προβλημάτων επεξεργασίας δεδομένων. Επίσης μελετήθηκαν αρκετές παρανοήσεις των μαθητών για την έννοια του πίνακα. Ένα σημαντικό εύρημα της έρευνας ήταν ότι οι παρανοήσεις των μαθητών για την έννοια της μεταβλητής ευθύνονται εν μέρει για κάποιες από τις παρανοήσεις τους στους πίνακες. Μια πιθανή εξήγηση για αυτό είναι ότι η μεταβλητή αποτελεί δομικό στοιχείο πάνω στο οποίο οικοδομείται η έννοια του πίνακα, οπότε είναι λογικό οι παρανοήσεις στην έννοια της μεταβλητής να μεταφέρονται ή να προκαλούν νέες παρανοήσεις στους πίνακες. Ένα άλλο σημαντικό εύρημα της έρευνας ήταν ότι πολλοί μαθητές έχουν σχηματίσει λανθασμένες αναπαραστάσεις για την έννοια του πίνακα και δεν αντιλαμβάνονται ότι ένα στοιχείο του πίνακα είναι ουσιαστικά μια μεταβλητή.

Από τα αποτελέσματα της έρευνας που έγινε για τη μελέτη των αναπαραστάσεων των μαθητών για τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής σε 168 μαθητές της Γ' τάξης τριών γενικών λυκείων και 53 πρωτοετών φοιτητών πληροφορικής προέκυψε ότι η πλειονότητα των μαθητών του δείγματος δυσκολεύονται στη διαχείριση παραλλαγών του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής. Στα προβλήματα με παραλλαγές του αλγορίθμου ευθείας ανταλλαγής, λίγοι μαθητές (κάτω από 20%) κατάφεραν να δώσουν απαντήσεις που κατατάσσονται στο συνθετικό επίπεδο της ταξινομίας SOLO. Ένας σημαντικός παράγοντας δυσκολίας ήταν η διαχείριση των εμφωλευμένων επαναλήψεων και των μεταβλητών τους (i,j), οι οποίες έχουν διττό ρόλο, το ρόλο του μετρητή στην εντολή επανάληψης αλλά και το ρόλο του δείκτη στοιχείου του πίνακα.

Τα παραπάνω αποτελέσματα αξιοποιήθηκαν για τον σχεδιασμό του εκπαιδευτικού περιβάλλοντος οπτικοποίησης αλγορίθμων DAVE. Υιοθετήθηκε η γραφική απεικόνιση του πίνακα ως μια ακολουθία διατεταγμένων κελιών τα οποία όμως δεν είναι ενωμένα μεταξύ τους, ώστε να φαίνεται ότι πρόκειται για ξεχωριστές μεταβλητές. Σε αυτό βοηθάει επίσης η κίνηση μεμονωμένων στοιχείων του πίνακα κατά την οπτικοποίηση της εκτέλεσης των αλγορίθμων ταξινόμησης. Επίσης, η αύξηση κάθε μετρητή της επανάληψης απεικονίζεται γραφικά με την κίνηση ενός βέλους το οποίο δείχνει στο κελί του πίνακα που δείχνει ο μετρητής με τη δεύτερη ιδιότητά του αυτή του δείκτη σε στοιχείο του πίνακα. Η οπτικοποίηση των μετρητών παίζει σημαντικό ρόλο στην κατανόηση της λειτουργίας των αλγορίθμων ταξινόμησης και αναζήτησης διότι στις περισσότερες περιπτώσεις χρησιμοποιείται εντολή επανάληψης στην οποία δεν φαίνεται η αύξηση του μετρητή.

Το λογισμικό που αναπτύχθηκε υιοθετεί διαφορετικά είδη οπτικοποίησης για κάθε αλγόριθμο έτσι ώστε να αναδεικνύονται τα ιδιαίτερα χαρακτηριστικά του. Για παράδειγμα η δυαδική αναζήτηση έχει αρκετά διαφορετική οπτικοποίηση από έναν αλγόριθμο ταξινόμησης. Το λογισμικό χρειάζεται για την εκτέλεσή του μόνο ένα πρόγραμμα πλοήγησης στον ιστό (browser) αφού έχει αναπτυχθεί με τεχνολογίες HTML5/Javascript, που σημαίνει ότι μπορεί να εκτελεστεί σε οποιαδήποτε πλατφόρμα, ακόμα και σε κινητό τηλέφωνο. Το σύστημα υποστηρίζει όλους τους γνωστούς αλγορίθμους ταξινόμησης και αναζήτησης που διδάσκονται στην δευτεροβάθμια εκπαίδευση και επιτρέπει ακόμα και την τροποποίηση του κώδικα των αλγορίθμων σε

κάποιες περιπτώσεις, επιτυγχάνοντας τον μέγιστο βαθμό διαδραστικότητας με τον χρήστη.

Η κύρια έρευνα αφορούσε τη μελέτη της συμβολής του, όσον αφορά τη διδακτική αξιοποίησή και τον τρόπο με τον οποίο αυτό μπορεί να υποστηρίξει την μαθησιακή διαδικασία. Τα αποτελέσματα της έρευνας έδειξαν ότι το DAVE μπορεί να ενταχθεί στη διδακτική και μαθησιακή πρακτική της τάξης με ενθαρρυντικά αποτελέσματα για τους μαθητές. Οι δυνατότητες πειραματισμού με την οπτικοποίηση της εκτέλεσης του αλγορίθμου, ενθάρρυναν τη συμμετοχή και την ενεργοποίηση των μαθητών κατά την επίλυση προβλημάτων ταξινόμησης πινάκων αυξημένης δυσκολίας.

Η παρούσα διατριβή ανέδειξε σημαντικές πληροφορίες για τη συμβολή του λογισμικού οπτικοποίησης αλγορίθμων ταξινόμησης πινάκων στην εξέλιξη της αλγοριθμικής σκέψης των μαθητών. Η χρήση του DAVE συνέβαλε ώστε οι μαθητές να αξιοποιήσουν αυτόνομα σημαντικές διαδικασίες στον προγραμματισμό, όπως ο εντοπισμός λογικών σφαλμάτων, η διόρθωση εντολών ή τμημάτων αλγορίθμων, η εμπειρική ανάλυση της απόδοσης αλγορίθμων και η ανάδειξη παρανοήσεων, ώστε να φτάσουν τελικά στην οικοδόμηση αλγόριθμων ταξινόμησης.

Τα αποτελέσματα της διατριβής αυτής επιβεβαιώνουν τα ευρήματα προηγούμενων ερευνών για τη συμβολή των περιβαλλόντων δυναμικής οπτικοποίησης αλγορίθμων στην ανάπτυξη προγραμματιστικών ικανοτήτων (Sorva et al., 2013· Urquiza–Fuentes et al., 2009). Η χρήση του λογισμικού DAVE ενίσχυσε την προσπάθεια των μαθητών του δείγματος για επίλυση αλγοριθμικών προβλημάτων με πίνακες, προωθώντας την ενεργό συμμετοχή και τον πειραματισμό τους με τις οπτικοποιήσεις αλγορίθμων. Οι μαθητές αξιοποίησαν σχεδόν όλες τις δυνατότητες του λογισμικού και, κυρίως, την τροποποίηση του κώδικα του αλγορίθμου και την αυξομείωση της ταχύτητας της εκτέλεσης της οπτικοποίησης. Τα δυναμικά χαρακτηριστικά του λογισμικού και οι δυνατότητες εντοπισμού λογικών λαθών που υποστηρίζει το λογισμικό, όπως η προσπέλαση πέρα από τα όρια του πίνακα, συνέβαλαν στην ανάδειξη λογικών σφαλμάτων και στην οικοδόμηση επαρκών αναπαραστάσεων των μαθητών για τις έννοιες του πίνακα, του δείκτη, της αντιμετάθεσης και της σύγκρισης στοιχείων.

Το λογισμικό μπορεί επίσης να συμβάλλει στην εμπειρική μελέτη της απόδοσης των αλγορίθμων ταξινόμησης και αναζήτησης, αξιοποιώντας τη δυνατότητα επιλογής των δεδομένων εισόδου, ώστε να διερευνηθεί η συμπεριφορά των αλγορίθμων σε ειδικές περιπτώσεις.

## *9.2 Σύγκριση με αντίστοιχες έρευνες και περιβάλλοντα οπτικοποίησης*

Τα τελευταία χρόνια τα σύγχρονα περιβάλλοντα οπτικοποίησης που έχουν αναπτυχθεί, επιτρέπουν την εκτέλεση των οπτικοποιήσεων στον ιστό, αλλά όχι την τροποποίηση του κώδικα του αλγορίθμου. Αυτό έχει ως αποτέλεσμα να περιορίζουν τις δυνατότητες πειραματισμού που έχει ο μαθητής με το λογισμικό και επομένως την αποτελεσματικότητα των τεχνικών διερευνητικής μάθησης που μπορούν να ακολουθηθούν.

Οι περισσότερες έρευνες που έχουν γίνει για την αξιολόγηση λογισμικών οπτικοποίησης καταλήγουν στο συμπέρασμα ότι δεν έχει σημασία η ποιότητα των γραφικών της οπτικοποίησης αλλά ο βαθμός αλληλεπίδρασης του μαθητή με το λογισμικό (Cetin & Andrews-Larson, 2016· Hundhausen, Douglas & Stasko, 2002· Shi, Min & Zhang, 2017· Sorva et al., 2013· Moreno et.al., 2004· Velazquez-Iturbide, Hernan-Losada, & Paredes-Velasco, 2017). Το λογισμικό πρέπει να δίνει τη δυνατότητα στον μαθητή να ορίσει τα δικά του δεδομένα να σταματάει και να ξεκινάει την οπτικοποίηση κατά βούληση ή να αυξομειώνει την ταχύτητά της.

Το σημαντικότερο πλεονέκτημα του DAVE είναι η δυνατότητα του μαθητή για τροποποίηση του κώδικα του αλγορίθμου σε ένα συγκεκριμένο εύρος. Μέχρι σήμερα το μόνο λογισμικό που δίνει αυτή τη δυνατότητα έστω και περιορισμένα είναι το DAVE και το PyAlgoViz (Laffra, 2014), το οποίο όμως εκτελεί τον αλγόριθμο στον διακομιστή και στη συνέχεια στέλνει το αποτέλεσμα της οπτικοποίησης στον υπολογιστή του χρήστη. Θα πρέπει να σημειωθεί όμως ότι το DAVE επειδή εκτελείται εξολοκλήρου στον υπολογιστή του χρήστη παρέχει υψηλότερου βαθμού διαδραστικότητα.

Τα αποτελέσματα της διατριβής επιβεβαιώνουν αντίστοιχα ευρήματα της βιβλιογραφίας σχετικά με τη συμβολή των λογισμικών οπτικοποίησης στη μαθησιακή διαδικασία του προγραμματισμού και της αλγοριθμικής (Boticki et al., 2013· Cetin & Andrews-Larson, 2016· Geller & Dios, 1998· Hansen et al., 2000). Η ενεργός εμπλοκή των μαθητών στην οπτικοποίηση των αλγορίθμων, ο βαθμός διαδραστικότητας και η επιλογή κατάλληλων αναπαραστάσεων για τις αλγοριθμικές δομές είναι τα πιο σημαντικά χαρακτηριστικά, τα οποία καθορίζουν την αποτελεσματικότητα ενός συστήματος οπτικοποίησης αλγορίθμων.

Ακολουθώντας στρατηγικές διερευνητικής μάθησης, και αξιοποιώντας την προσομοίωση του αλγορίθμου μέσω του λογισμικού DAVE, οι μαθητές είχαν την

ευκαιρία να δοκιμάσουν την δική τους εκδοχή του αλγορίθμου επίλυσης του προβλήματος και να πειραματιστούν με αυτήν, εντοπίζοντας τυχόν λογικά λάθη μέσα από την οπτικοποίηση.

## *9.3 Αξιοποίηση του DAVE στη εκπαιδευτική πρακτική*

Το λογισμικό DAVE μπορεί να αξιοποιηθεί στην δευτεροβάθμια και στην τριτοβάθμια εκπαίδευση για την παρουσίαση και επεξήγηση των βασικών αλγορίθμων ταξινόμησης και αναζήτησης. Το σύστημα είναι σχεδιασμένο ώστε να υποστηρίζει τη διερευνητική μάθηση των μαθητών με στόχο την κατανόηση της λειτουργίας του αλγορίθμου.

Το λογισμικό υποστηρίζει όλους τους αλγόριθμους ταξινόμησης (εισαγωγής, επιλογής, ευθείας ανταλλαγής) και αναζήτησης (σειριακή, δυαδική) που διδάσκονται στα μαθήματα πληροφορικής στην δευτεροβάθμια εκπαίδευση. Μπορεί όμως να χρησιμοποιηθεί και στην τριτοβάθμια εκπαίδευση σε φοιτητές πληροφορικής στο πλαίσιο μαθημάτων ανάλυσης της πολυπλοκότητας των αλγορίθμων ταξινόμησης και αναζήτησης. Οι μαθητές μπορούν να εκτελέσουν ένα σχέδιο εμπειρικής ανάλυσης της πολυπλοκότητας ενός αλγορίθμου δοκιμάζοντας διάφορες κατηγορίες δεδομένων εισόδου για κάθε αλγόριθμο.

Ο εκπαιδευτικός καλείται να εκμεταλλευτεί τα ιδιαίτερα χαρακτηριστικά του λογισμικού μέσα από κατάλληλα σχεδιασμένες δραστηριότητες που προάγουν τον πειραματισμό των μαθητών με την οπτικοποίηση του αλγορίθμου. Οι μαθησιακές δραστηριότητες θα πρέπει να βασίζονται στην παιδαγωγική αξιοποίηση του λογικού σφάλματος και των διαφορών ανάμεσα στα αναμενόμενα και στα παρατηρούμενα αποτελέσματα, με στόχο την οικοδόμηση αποτελεσματικών αναπαραστάσεων για τις αλγοριθμικές δομές και τα προγραμματιστικά αντικείμενα.

Αξιοποιώντας τα ευρήματα της παρούσας διατριβής, πρόκειται να σχεδιάσουμε μια πλήρη ακολουθία μαθησιακών δραστηριοτήτων και εκπαιδευτικών σεναρίων για όλους τους αλγορίθμους που υποστηρίζει το DAVE. Το σχετικό υλικό θα είναι ελεύθερα διαθέσιμο για τους εκπαιδευτικούς.

## *9.4 Προτάσεις για παραπέρα έρευνα*

Οι περιορισμοί της τελικής έρευνας της παρούσας διατριβής σχετίζονται με το δείγμα που επιλέξαμε και την περιορισμένη χρήση του DAVE στο εργαστήριο για μια συγκεκριμένη ομάδα αλγορίθμων. Θα είχε ενδιαφέρον η μελέτη της εξέλιξης της

αλγοριθμικής σκέψης των μαθητών σε μια πλήρη σειρά μαθημάτων για αλγορίθμους επεξεργασίας πινάκων με χρήση του DAVE ως διερευνητικό-εποικοδομητικό εργαλείο.

Θα είχε ενδιαφέρον μια μελλοντική έρευνα σε δυο ομάδες μαθητών μια πειραματική ομάδα και μια ομάδα ελέγχου. Στην ομάδα ελέγχου οι αλγόριθμοι ταξινόμησης θα παρουσιάζονται χωρίς τη χρήση του λογισμικού οπτικοποίησης DAVE. Οι μαθητές της πειραματικής ομάδας θα χρησιμοποιήσουν το DAVE μέσα από κατάλληλους διερευνητικούς σχεδιασμούς, ώστε να ανακαλύψουν τα ιδιαίτερα χαρακτηριστικά του αλγορίθμου. Η συγκριτική μελέτη των αποτελεσμάτων των δύο ομάδων μπορεί να δώσει περισσότερες πληροφορίες για το βαθμό στον οποίο η οπτικοποίηση αλγορίθμων συμβάλλει στην οικοδόμηση προγραμματιστικών εννοιών από τους μαθητές για ένα εύρος αλγορίθμων.

Επίσης θα είχε ερευνητικό ενδιαφέρον η ποιοτική μελέτη (π.χ. μέσω συνεντεύξεων) των απόψεων των εκπαιδευτικών οι οποίοι έχουν εντάξει συστηματικά το DAVE στη διδασκαλία τους. Η αποτίμηση της συστηματικής χρήσης του DAVE, μπορεί να δώσει σημαντική ανατροφοδότηση, τόσο για το σχεδιασμό εκπαιδυτικών παρεμβάσεων με αυτό όσο και για τη βελτίωση ή την επέκταση της λειτουργικότητάς του.

Τέλος, σχεδιάζουμε την επέκταση του λογισμικού, ώστε να επιτρέπει την οπτικοποίηση βασικών αλγοριθμικών εννοιών και άλλων κατηγοριών αλγορίθμων. Για παράδειγμα, οπτικοποίηση αλγορίθμων αναδρομής και αλγόριθμων με γράφους. Παράλληλα σχεδιάζουμε να επεκτείνουμε τις δυνατότητες τροποποίησης του κώδικα του αλγορίθμου από τον μαθητή-φοιτητή, ώστε να ενισχυθούν ακόμη περισσότερο τα διερευνητικά και εποικοδομητικά χαρακτηριστικά του DAVE.

# *Δημοσιευθείσες Εργασίες*


1. Vrachnos, E., & Jimoyiannis, A. (2017). Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy. *Themes in Science and Technology Education,* 10(1), 31-52.
2. Vrachnos, E., & Jimoyiannis, A. (2014). Design and evaluation of a web-based dynamic algorithm visualization environment for novices. *Procedia Computer Science Journal,* 27, pp. 229-239.
3. Vrachnos, E., & Jimoyiannis, A. (2008). Dave: A Dynamic Algorithm Visualization Environment for Novice Learners. *8th IEEE International Conference on Advanced Learning Technologies* (pp. 319-323). Santander: IEEE Computer Society.
4. Βραχνός, Ε., & Τζιμογιάννης, Α. (2018). Προϋπάρχουσες γνώσεις των μαθητών Γυμνασίου και Λυκείου στους αλγόριθμους ταξινόμησης. Μια συγκριτική ανάλυση. *Πρακτικά 9ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Θεσσαλονίκη.
5. Βραχνός, Ε., & Τζιμογιάννης, Α. (2018). Σχεδιασμός και Ανάπτυξη ενός Εκπαιδευτικού Περιβάλλοντος Δυναμικής Οπτικοποίησης Αλγορίθμων. *Πρακτικά 9ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Θεσσαλονίκη.
6. Βραχνός, Ε., & Τζιμογιάννης, Α. (2016). Προϋπάρχουσες γνώσεις των μαθητών στους αλγόριθμους ταξινόμησης. Ποιον αλγόριθμο επινοούν; *Πρακτικά 8ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Ιωάννινα, σελ. 61-68.
7. Βραχνός, Ε., & Τζιμογιάννης, Α. (2014). Μελέτη της συμβολής του περιβάλλοντος οπτικοποίησης αλγορίθμων DAVE στην οικοδόμηση αλγορίθμων ταξινόμησης από μαθητές Γ' Λυκείου. *Πρακτικά 7ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Ρέθυμνο.
8. Βραχνός, Ε., & Τζιμογιάννης, Α. (2014). Αναπαραστάσεις μαθητών και φοιτητών για τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής: Μια ανάλυση βασισμένη στην ταξινομία SOLO. *Πρακτικά 7ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Ρέθυμνο.

9. Βραχνός, Ε., & Τζιμογιάννης, Α. (2010). Μελέτη των αναπαραστάσεων μαθητών της Γ' Λυκείου για την έννοια του πίνακα χρησιμοποιώντας την ταξινομία SOLO. *Πρακτικά 5ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"* (σ. 81-90). Αθήνα.
10. Βραχνός, Ε. & Τζιμογιάννης, Α. (2009). Εκπαιδευτικά περιβάλλοντα οπτικοποίησης αλγορίθμων: Μια επισκόπηση των τεχνικών και παιδαγωγικών χαρακτηριστικών. *Θέματα Επιστημών και Τεχνολογίας στην Εκπαίδευση* Τομ. 2, Αρ. 3, σελ. 215-245.
11. Βραχνός, Ε., & Τζιμογιάννης, Α. (2008). Σχεδιασμός ενός Περιβάλλοντος Δυναμικής Οπτικοποίησης Αλγορίθμων: Το σύστημα DAVE, *Πρακτικά 4ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής",* Εκδόσεις Νέων Τεχνολογιών, σελ.171-180.

# *Βιβλιογραφία*


Adcock, B., Bucci, P., Heym, D. W., Hollingsworth, E. J., Long, T., & Weide, B. (2007). Which pointers errors do students make? *SIGCSE Bulletin, 39*(1), 9-13.

Aleksic, V. & Ivanovic, M. (2016). Introductory programming subject in european higher education. *Informatics in Education*, 15(2), 163-182.

Anderson, L. W. and Krathwohl, D. A. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York: Longman.

Annamaa, A. (2015). Introducing Thonny, a Python IDE for learning programming. *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. ACM, New York, NY, USA, 117-121

Astrachan, O. & Briggs, A. (2012). The CS principles project. *ACM Inroads, 3*(2), 38-42.

Astrachan, O. (2003). Bubble sort: an archaeological algorithmic analysis. *Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03)*. (pp. 1-5). NY: ACM.

Baecker, R. M. (1981). Sorting out Sorting. Narrated color videotape presented at ACM SIGGRAPH ’81.

Balanskat, A., Engelhardt, K. (2014). Computing our future: Computer programming and coding-priorities, school curricula and initiatives across Europe. European Schoolnet.

Bau, D., Bau, A., Dawson, M., and Pickens, S. (2015). Pencil code: block code for a text world. *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 445-448.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: blocks and beyond. *Communications of the ACM, 60*(6), 72-80.

Bayman, P., and Mayer, R. E. (1983). A diagnosis of beginning programmers’ misconceptions of BASIC programming statements. *Communications of the ACM*, 26(9), 677–679.

Bell, T., Andreae, P., & Lambert, L. (2010). Computer Science in New Zealand high schools. In *Proceedings of the Twelfth Australasian Conference on Computing Education* - Volume 103. Australian Computer Society, Inc, Brisbane, Australia, 15–22.

Benander, C. A., Benander, A. B., & Howard Pu. (1996). Recursion vs. iteration: an empirical study of comprehension. *Journal of Systems and Software, 32*(1), 73-82.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching, 20*(1):45-73.

Ben-Bassat Levy, R., & Ben-Ari., M. (2009). A survey of research on the jeliot program animation system. Chais Conference on Instructional Technologies Research: Learning in the Technological Era. Raanana. Israel: The Open University of Israel.

Bennedsen, J. & Caspersen, E. M. (2007). Failure rates in introductory programming. *SIGCSE Bulletin. 39*(2), 32-36.

Biggs, J. B., & Collis, K. F. (1982). *Evaluating the quality of learning. The SOLO taxonomy.* New York: Academic Press.

Blanchard, J. (2017). Hybrid Environments: A Bridge from Blocks to Text. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM, New York, NY, USA, 295-296.

Bloom, B. S. (1956). *Taxonomy of educational objectives, Handbook 1: cognitive domain*. New York: Addison Wesley.

Bonar, J. & Soloway, E. (1985). Preprogramming knowledge: a major source of misconceptions in novice programmers. *Human-Computer Interaction, 1*(2), 133- 161.

Bonifaci, V., Demetrescu, C., Finocchi, I., & Laura L. (2006). Visual editing of animated algorithms: The Leonardo Web Builder. *Proceedings of the Working Conference on Advanced Visual Interfaces* (pp. 476-479), Venezia, Italy, ACM.

Bontá, P., Papert, A., & Silverman, B. (2010). Turtle, art, turtleart. *Proceedings of Constructionism 2010 Conference*.

Boticki, I., Barisic, A., Martin, S., & Drljevic, N. (2013). Teaching and learning computer science sorting algorithms with mobile devices: A case study. *Computer Applications in Engineering Education, 21*(S1), E41-E50.

Boustedt, J. (2012). Students' different understandings of class diagrams. *Computer Science Education,* 22(1), 29–62.

Brown, M.H. (1991). Zeus: A system for algorithm animation and multi-view editing. 1991 IEEE Workshop on Visual Languages (pp. 4-9). Kobe, Japan.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing Computer Science Back into Schools: Lessons from the UK. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education. SIGCSE '13*. ACM, New York, NY, USA, 269–274.

Byckling, P. & Sajaniemi, J. (2006). Roles of variables and programming skills improvement. *ACM SIGCSE Bulletin, 38*(1), 413-417.

Campbell, J., Smith, D., & Brooker, R. (1998). From conception to performance: how undergraduate students conceptualise and construct essays. *Higher Education, 36*(4), 449-469.

Carlson, M. P. (1998). A cross-sectional investigation of the development of the function concept. In A.H. Schoenfeld, J. Kaput, & E. Dubinsky (Eds.), *Research in Collegiate Mathematics Education. III. CBMS Issues in Mathematics Education* (pp. 114-162). Providence, RI: American Mathematical Society.

Carmer, B., Rosulek, M., (2015). Vamonos: Embeddable visualizations of advanced algorithms, IEEE Frontiers in Education Conference, 1–8.

Caspersen, E. M., Gal-Ezer, J, McGettrick, A., & Nardelli, E. (2018). Informatics for all the Strategy. Technical Report. ACM, New York.

Cetin, H., & Andrews-Larson, C. (2016). Learning sorting algorithms through visualization construction. *Computer Science Education, 26*(1), 27-43.

Cetin, I. (2015). Students' Understanding of Loops and Nested Loops in Computer Programming: An APOS Theory Perspective. *Canadian Journal of Science Mathematics and Technology Education*, 15(2), 155-170.

Charlton J. P. & Birkett P. E. (1994). Specificity Versus Nonspecificity of Cognitive Skills in Elementary Computer Programming. *Journal of Research on Computing in Education, 26*(3), 391-402.

Chen, T.-Y., Lewandowski, G., McCartney, R., Sanders, K., and Simon, B. (2007). Commonsense computing: using student sorting abilities to improve instruction. *SIGCSE Bulletin. 39*(1), 276-280.

Chick, H. (1998). Cognition in the formal modes: research mathematics and the SOLO taxonomy. *Mathematics Education Research Journal, 10*(2), 4-26.

Clear, T., Whalley, J., Lister, R., Carbone, A., Hü, M., Sheard, J., Simon, B., & Thomson, E. (2008). Reliably classifying novice programmer exam responses using the SOLO taxonomy. In S. Mann & M. Lopez (Eds.), *Proceedings of the 21st Annual Conference of the National Advisory Committee on Computing Qualifications* (pp. 23-30). Auckland: NACCQ.

CodeMirror. (2018, May 30). Retrieved 12 June 2018 from https://codemirror.net/ .

Cohen, M. (2013). Uncoupling Alice: using Alice to teach advanced object-oriented design. *ACM Inroads, 4*(3), 82-88.

College Board (2017). Computer Science: Principles, Big Ideas, Key Concepts and Supporting Concepts, Retrieved 12 June 2018 from https://apcentral.collegeboard.org/

Corney, M., Fitzgerald, S., Hanks, B., Lister, R., McCauley, R., & Murphy, L. (2014). 'explain in plain english' questions revisited: data structures problems. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 591-596). New York: ACM Press.

Corney, M., Lister, P. & Teague, D. (2011). Early relational reasoning and the novice programmer: swapping as the "hello world" of relational reasoning. *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114 (ACE '11)*, John Hamer and Michael de Raadt (Eds.), Vol. 114. pages: 95-104.

Craig, M., & Petersen, M. (2016). Student difficulties with pointer concepts in C. *Proceedings of the Australasian Computer Science Week Multiconference* (Art. 8). New York: ACM.

Crescenzi, P., & Nocentini, C. (2007). Fully Integrating Algorithm Visualization into a CS2 course. *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education* (pp.296-300). Dundee, Scotland: ACM.

Cross, J. H., & Hendrix, T. D. (2006). jGrasp: A lightweight IDE with dynamic object viewers for CS1 and CS2. *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education*, (pp.356-356). Bologna, Italy: ACM.

Dagienė, V. (2008). Teaching Information Technology and Elements of Informatics in Lower Secondary Schools: Curricula, Didactic Provision and Implementation. *Proceedings of the 3rd international conference on Informatics in Secondary Schools - Evolution and Perspectives: Informatics Education - Supporting Computational Thinking (ISSEP '08)*, Roland T. Mittermeir and Maciej M. Sysło (Eds.). Springer-Verlag, Berlin, Heidelberg, 293-304.

Dale, N. B. (2006). Most difficult topics in CS1: results of an online survey of educators. *SIGCSE Bulletin*. *38*(2), 49-53.

Danielsiek, H., Wolfgang, P., & Vahrenhold, J. (2012). Detecting and understanding student's misconceptions related to algorithms and data structures. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (SIGCSE 2012), Raleigh, North Carolina (pp. 21-26). New York: ACM.

de Raadt, M. (2007). A review of Australian investigations into problem solving and the novice programmer. *Computer Science Education*, *17*(3), 201-213.

du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 57-73.

Du, J., Wimmer, H., & Rada, R. (2018). "Hour of Code": A Case Study. *Information Systems Education Journal*, 16(1), 51-60.

Eckerdal, A., & Thune, M. (2005). Novice java programmers' conceptions of "object" and "class", and variation theory. *ACM SIGCSE Bulletin*, *37*(3), 89-93.

ECS, Exploring Computer Science (2017), Retrieved 12 June from http://www.exploringcs.org

Ennis D. L. (1994). Computing Problem-Solving Instruction and Programming Instruction to Increase the Problem-Solving Ability of High School Students, *Journal of Research on Computing in Education*, 26(4), 489-496.

Flannery, P., L., Silverman, B., Kazakoff, R. E., Umaschi Bers, M., Bontá, P., and Resnick, M. (2013). Designing ScratchJr: support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, USA, 1-10.

Fleury, A. E. (1991). Parameter passing: the rules the students construct. *ACM SIGCSE Bulletin*, *23*(1), 283-286.

Fleury, A. E. (2000). Programming in java: student-constructed rules. *ACM SIGCSE Bulletin*, *32*(1), 197-201.

Forbes, J. (2012). The CS 10K Project: mobilizing the computing community around high *school education Journal of Computing Sciences in Colleges 28(1)*, 5-6.

Fraser, N. (2015). Ten things we've learned from Blockly. *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond*. IEEE Computer Society, Washington, DC, USA, 49-50.

Furber, S. (2012). Shutdown or Restart. The way forward for computing in UK schools. *London: The Royal Society*.

Gal-Ezer, J. & Chris Stephenson, C. (2014). A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States. *ACM Transactions on Computing Education, 14*(2), 18 pages.

Gal-Ezer, J., Beeri, C., Harel, D., and Yehudai, A. (1995). A High School Program in Computer Science. *Computer*, *28*(10), 73–80.

Garcia, D. D., Astrachan, O., Brown, B., Gray, J., Lin, C., Beth, B., Morelli, R., desJardins, M., & Sridhar, N. (2015). Computer Science Principles Curricula: On-the-ground; adoptable; adaptable; approaches to teaching. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 176-177.

Geller, J., & Dios, R. (1998). A low-tech, hands-on approach to teaching sorting algorithms to working students, *Computers & Education, 31*(1), 89-103.

Gestwicki, P., & Jayaraman, B. (2002). Interactive visualization of Java programs. *IEEE 2002 Symposia on Human Centric Computing Languages and Environments* (pp. 226-235). Arlington, USA: IEEE Computer Society.

Ginat, D & Menashe, E. (2015). SOLO Taxonomy for Assessing Novices' Algorithmic Design. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (SIGCSE '15). ACM, New York, NY, USA, 452-457

Goode, J. and Margolis, J. (2011). Exploring Computer Science: A Case Study of School Reform. *ACM Transactions on Computing Education, 11*(2), 12:1-12:16.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: an evaluation of the educational game light-bot. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (ITiCSE '13). ACM, New York, NY, USA, 10-15.

Gries, D. (2008). A principled approach to teaching OO first. *Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08)*. ACM, New York, NY, USA, 31-35.

Grissom, S., Murphy, L., McCauley, R., & Fitzgerald, S. (2016). Paper vs. Computer-based Exams: A Study of Errors in Recursive Binary Tree Algorithms. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16).* ACM, New York, NY, USA, 6-11.

Guibas, L., & Sedgewick, R. (1978) A dichromatic framework for balanced trees. IEEE Symposium on Foundations of Computer Science (pp. 8-21). Michigan: IEEE Computer Society.

Guo J. P. (2012). Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. *Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education*, ACM, New York, pp. 579-584.

Hadas, L., R. (2013). A derivation-first approach to teaching algorithms. *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)* (pp. 573-578). NY: ACM.

Halim, S. (2015). VisuAlgo – Visualizing Data Structures and Algorithms Through Animation, *Olympiads in Informatics*, 9, 243–245.

Hansen, S, Narayanant, H, & Hegarty, M. (2002). Designing educationally effective algorithm visualizations, *Journal of Visual Languages & Computing, 13*(3), 291–317.

Harvey, B. and Mönig, J. (2010). Bringing "no ceiling" to Scratch: Can one language serve kids and computer scientists? In J. Clayson & I. Kalas, eds. *Proceedings of Constructionism 2010 Conference*. 1–10.

Hazzan, O., Lapidot, T., & Ragonis, N. (2011). *Guide to teaching computer science: an activity-based approach*. London: Springer.

Helminen, J., & Malmi, L. (2010). Jype - a program visualization and programming exercise tool for Python. *Proceedings of the 5th international symposium on Software visualization* (SOFTVIS '10). ACM, New York, NY, USA, 153-162.

Hoc, J.-M., Green, T. R. G., Samurçay, R., & Gilmore D. J. (1990). *Psychology of Programming*. London: Academic Press.

Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding object misconceptions. *ACM SIGCSE Bulletin, 29*(1), 131-134.

Horn, S., M., Brady, C., Hjorth, A., Wagh, A., and Wilensky, U. (2014). Frog pond: a code first learning environment on evolution and natural selection. *Proceedings of the 2014 conference on Interaction design and children (IDC '14)*.(pp. 357-360) New York:ACM,.

Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and correcting java programming errors for introductory computer science students. *ACM SIGCSE Bulletin, 35*(1), 153-156.

Hubwieser, P. (2012). Computer Science Education in Secondary Schools - The Introduction of a New Compulsory Subject. *ACM Transactions on Computing Education*. 12(4), 16:1- 16:41.

Hundhausen, C., & Brown, J. L. (2007). What you see is what you code: a 'live' algorithm development and visualization environment for novice learners. *Journal of Visual Languages and Computing, 18*(1), 22-47.

Hundhausen, C., & Douglas, S. (2002) Low-fidelity algorithm visualization. *Journal of Visual Languages and Computing, 13*(5), 449-470.

Hundhausen, C., Douglas, S., & Stasko, J. (2002). A metastudy of algorithm visualization effectiveness. *Journal of Visual Languages and Computing, 3*(3), 259-290.

Izu, C., Weerasinghe, A., & Pope, C. (2016). A study of code design skills in novice programmers using the SOLO taxonomy. *Proceedings of the 2016 ACM Conference on International Computing Education Research* (pp. 251-259). New York: ACM.

Jakwerth, P., & Stancavage, F. (2003). *An investigation of why students do not respond to questions*. Working Paper No. 2003-12. Washington, DC: US Department of Education, National Center for Education Statistics.

Jimoyiannis A., Christopoulou E., Paliouras A., Petsos A., Saridaki A., Toukiloglou P., & Tsakonas P. (2013). Design and development of learning objects for lower secondary education in Greece: The case of computer science e-books. *Proceedings of EDULEARN13 Conference* (pp. 41-49). Barcelona: IATED.

Jimoyiannis, A. (2011). Using SOLO taxonomy to explore students' mental models of the programming variable and the assignment statement. *Themes in Science and Technology Education*, *4*(2), 53-74.

Jimoyiannis A. (2009). Computer simulations and scientific knowledge construction, in A. Cartelli & M. Palma (eds.), Encyclopedia of Information Communication Technology (pp. 106–120). Hershey, PA: IGI Global.

Johnson, C., J., & Fuller, U. (2006). Is Bloom's taxonomy appropriate for computer science? *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006* (pp. 120-123). New York: ACM.

Jonas, M. & Sabin, M. (2015). Computational thinking in Greenfoot: AI game strategies for CS1: conference workshop. *Journal of Computing Sciences in Colleges*, *30*(6), 8-10.

Joosten, S., van den Berg, K., & van der Hoeven, G. (1993). Teaching functional programming to first-year students. *Journal of functional programming*, *3*(1), 49-65.

Joy, M. & Matthews, S. (2006). Some experiences in teaching functional programming, *International Journal of Mathematical Education in Science and Technology*, 25(2), 165-172.

JSamba (1998). http://www.cs.gatech.edu/gvu /softviz/algoanim/jsamba.

K-12 Computer Science Framework (2016). https://k12cs.org/

Kagan D. M. (1989). Research on Computer Programming as a Cognitive Activity: implications for the study of classroom teaching, *Journal of Education for Teaching*, *15*(3), 177-189.

Karavirta, V., Korhonen, A., Malmi, L., & Stalnacke, K. (2004). MatrixPro. A tool for on-the-fly demonstration of data structures and algorithms. *Third Program Visualization Workshop* (pp. 26-33). The University of Warwick, UK.

Karpierz, K., & Wolfman, S. (2014). Misconceptions and concept inventory questions for binary search trees and hash tables. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. Atlanta, Georgia (pp. 109-114). New York: ACM.

Kessler, M. C. & Anderson, R. J. (1986). Learning Flow of Control: Recursive and Iterative Procedures, *Human–Computer Interaction*, *2*(2), 135-166.

Klopfer, E., Scheintaub, H., Huang, W, Wendel, D., Roque, R. (2009). The Simulation Cycle - Combining Games, Simulations, Engineering and Science Using StarLogo TNG. *Journal of E-Learning and Digital Media*, *6*(1), 71-96.

Knuth, E. D. (1998). *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Kölling, M., Quig, B., Patterson, A. & Rosenberg, J. (2003). The BlueJ system and its pedagogy, *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13*(4), 249-268.

Korhonen, A., & Malmi, L. (2002). Matrix - Concept animation and algorithm simulation system. Working Conference on Advanced Visual Interfaces (pp. 109-114). Trento, Italy: ACM.

Korhonen, A., Malmi, L., & Silvasti, P. (2003). TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. *Proceedings of the Third Annual Baltic Conference on Computer Science Education* (pp. 48-56) Joensuu, Finland: Institute of Mathematics and Informatics, Vilnius, Lithuanie.

Kumar, D. (2014). Digital playgrounds for early computing education. *ACM Inroads, 5(1)*, 20-21.

Kunkle, M., W., & Allen, B., R. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education*, Vol. 16(1), pages 3:1 - 3:26.

Laakso, M., Myller, N., & Korhonen, A. (2009). Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels. Educational Technology & Society, 12(2), 267–282.

Laffra, C. (2014). PyAlgoViz: Python algorithm visualization in the browser. Presentation given at the PYCON 2014, May 3, Montreal.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*. *37*(3), 14-18.

Lake, D. (1999). Helping students to go SOLO: teaching critical numeracy in the biological sciences. *Journal of Biological Education*, *33*(4), 191-198.

Lister, R., Adams, S., E., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, E. J., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin, 36*(4), 119-150.

Megalou, E., & Kaklamanis, C. (2014). Photodentro LOR, the Greek national learning object, In *Proceedings of INTED2014: 8th International Technology, Education and Development Conference*, Valencia(Spain), 1–11.

Lister, R., Clear, T., Simon, Bouvier, D., Carter, P., Eckerdal, A., Jackova, J., Lopez, M., McCartney, R., Robbins, P., Seppällä, O., & Thomson, E. (2009). Naturally occurring data as research instrument: analysing examination responses to study the novice programmer. *ACM SIGCSE Bulletin, 41*(4), 156–173.

Lister, R., Simon, B., Thomson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin, 38*(3), 118–122.

Liu, C.H., Jiu, Y.W., & Jason, J-Y. (2009). Using design sketch to teach bubble sort in high school. *The Journal of Computing, 1*(1), 20-25.

Lu, J. & Fletcher, G. (2009). Thinking about computational thinking. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (SIGCSE' 09).

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers, *Computer Science Education, 21*(1), 57-80.

Madison, A., & Gifford, J. (2003). Modular programming: novice misconceptions. *Journal of Research on Technology in Education, 34*(3), 217-219.

McCauley, R., Hanks, B., Fitzgerald, S., & Murphy, L. (2015). Recursion vs. Iteration: An Empirical Study of Comprehension Revisited. *Proceedings of the 46th ACM Technical* Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 350-355.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B-D., Cary Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*(4), 125-180.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. M. (2013). Learning computer science concepts with Scratch, *Computer Science Education*, *23*(3), 239-264.

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming-Views of Students and Tutors. *Education and Information Technologies, 7*(1), 55-66.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). Visualizing Program with Jeliot3. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI, (pp. 373-380). Bari, Italy: ACM.

Myers, B.A. Taxonomies of visual programming and program visualisation. (1990). *Journal of Visual Languages of Computing*, *1*(1), 97-123.

Myller, N., Bednarik, R., Sutinen, E., & Ben-Ari, M., (2009). Extending the engagement taxonomy: software visualization and collaborative learning. *ACM Transactions on Computing Education*, *9*(1), Article 7.

Naps T., Eagan J.R., & Norton L.L. (2003). JHAVE. An environment to actively engage students in Web-based algorithm visualizations. *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education* (pp. 109-113). Reno, Nevada, USA: ACM.

Naps, T., Roessling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velasquez Iturbide, J., A. (2003). Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin, 35*(2), 131-152.

Papert S. (1980) *Νοητικές Θύελλες: Παιδιά, ηλεκτρονικοί υπολογιστές και δυναμικές ιδέες*, Εκδόσεις Οδυσσέας.

Pappas, I., Giannakos, M., & Jaccheri, L. (2016). Investigating Factors Influencing Students' Intention to Dropout Computer Science Studies. *Proceedings of the 2016 ACM Conference on Innovation*

*and Technology in Computer Science Education (ITiCSE '16)*. ACM, New York, NY, USA, 198-203.

Pareja-Flores, C., Urquiza Fuentes J.A., & Velasquez Iturbide, J. (2007). WinHIPE: An IDE for functional programming based on rewriting and visualization, *ACM SIGPLAN Notices, 43*(2), 14-23.

Pausch, R., Burnette, T., Capeheart, A.C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S., & White, J. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications, 15*(3), 8-11.

Pea, R. D. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research 2*(1), 25–36.

Petersen, A., Craig, M., Campbell, J., & Tafliovich, A. (2016). Revisiting why students drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)*. ACM, New York, NY, USA, 71-80

Petersen, A. Graig, M., & Zingaro, D. (2011). Reviewing CS1 exam question content. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11*) (pp. 631-636). New York: ACM.

Pierson, W., & Rodger S. (1998). Web-based animation of data structures using JAWAA. *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '98) (pp. 267–271). Atlanta, Georgia, USA, ACM.

Price, B.A. Baecker, R. M., & I.S. Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing, 4*(3), 211-266.

Putnam, T.R., Sleeman, D., Baxter, J.A., & Kuspa, L.K. (1986). A summary of misconceptions of high school basic programmers. *Journal of Educational Computing Research*, 2(1), 459–472.

Ragonis, N., & Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education*, *15*(3), 213-221.

Rajala, T., Laakso, M., Kaila, E., & Salakoski, T. (2007). VILLE – A language-independent program visualization tool. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*, Vol.88, 151-159 Kolli National Park, Finland: ACS.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

Rinderknecht, C. (2014). A Survey on the Teaching and Learning of Recursive Programming. *Informatics in Education. 13*(1). 87-119.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172.

Rogalski, J. & He, Y. (1989). Logic abilities and mental representations of the informatical device in acquisition of conditional structures by 15-16 year old students. *European Journal of Psychology of Education*, **4**, 71-82.

Roman, G. C., & Cox K. C. (1993). A taxonomy for program visualisation systems. *Computer*, *26*(12), 11-24.

Rößling, G., & Freisleben, B. (2002). ANIMAL: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages & Computing*, *13*(3), 341-354.

Rössling, G., & Schroeder, P. (2009). Animalipse. An eclipse plugin for AnimalScript. *Electronic Notes in Theoretical Computer Science*, 224, 3-14.

Sajaniemi J. (2002). Visualizing Roles of Variables to Novice Programmers. *Proceedings of the Fourteenth Annual Workshop of the Psychology of Programming Interest Group*, (pp. 111-127). London, U.K.

Sajaniemi, J. & Kuittinen, M. (2005). An experiment on using roles of variables in teaching introductory programming. *Computer Science Education*, 15, 59–82.

Sajaniemi, J. (2005). Roles of variables and learning to program. Invited talk, In A. Jimoyannis (ed), *Proceedings of the 3rd Panhellenic Conference "Didactics of Informatics"*, University of Peloponnese, Korinthos, Greece.

Sajaniemi, J., & Kuittinen, M. (2003). Program animation based on the roles of variables. *ACM Symposium on Software Visualization*, San Diego, USA: ACM, 7-16.

Samba (1998). Retrieved 12 June 2018 from

https://www.cc.gatech.edu/gvu/ii/softvis/algoanim/samba.html

Samurçay, R. (1989). The concept of variable in programming: Its meaning and use in problem-solving by novice programmers. In *Studying the Novice Programmer*, E. Soloway and J. C. Spohrer (eds), Hillsdale, NJ: LEA, pp. 161-178.

Scratch (2010). http://scratch.mit.edu

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cuniff, D., Boucher Owens, B., Stephenson, C., & Verno, A. (2011). CSTA K-12 Computer Science Standards. Revised 2011. CSTA Standards Task Force. CSTA, New York.

Seiter, L. (2015). Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (SIGCSE '15). ACM, New York, NY, USA, 540-545

Seppälä, O., Malmi, L., & Korhonen A. (2006). Observations on student misconceptions – A case study of the Build-Heap algorithm. *Computer Science Education*, *16*(3), 241–255.

Sheard, J., Carbone, A., Lister, R., Simon, B., Thomson, E., & Whalley, J. L. (2008). Going SOLO to assess novice programmers. *ACM SIGCSE Bulletin*, *40*(3), 209-213.

Shi, N., Min, Z, & Zhang, P. (2017). Effects of visualizing roles of variables with animation and IDE in novice program construction. *Telemat. Inf. 34*(5), 743-754.

Shuhidan, S., Hamilton, M., & D' Souza, D. (2009). A taxonomic study of novice programming summative assessment. In M. Hamilton, & T. Clear (Eds.), *Proceedings of the 11th Australasian Conference on Computing Education (ACE '09)* (pp. 147-156). Darlinghurst: Australian Computer Society, Inc.

Simon, B., Chen, T.-Y., Lewandowski, G., McCartney, R., & Sanders, K. (2006). Commonsense computing: What students know before we teach (Episode 1): Sorting. *Proceedings of the 2nd International Workshop on Computing Education Research* (pp. 29-40). NY: ACM.

Sirkiä, T., & Sorva, J. (2012). Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. *Proceedings of the 12th Koli Calling International Conference on Computing Education Research* (pp. 19-28). New York: ACM.

Smith, N., Sutcliffe, C., Sandvik, L. (2014). Code club: bringing programming to UK primary schools through Scratch. Proceedings of the *45th ACM Technical Symposium on Computer Science Education*, pp. 517-522.

Soloway, E., & Spohrer, J.C. (1989). Studying the novice programmer. Hillsdale, NJ: Lawrence Erlbaum.

Sorva, J. & Sirkiä, T. (2010). UUhistle: a software tool for visual program simulation. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling '10)*. (pp. 49-54), ACM, New York.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, *13*(2), Article 8.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education 13*(4), Article 15.

Stasko, J. & Patterson, C. (1992). Understanding and characterizing software visualization systems. *IEEE Workshop on Visual Languages* (pp. 3-10). Seattle, USA: IEEE Computer Society.

Stasko, J. (1990). TANGO: A framework and system for algorithm animation. *IEEE Computer*, 23(9), 27-39.

Stasko, J. (1997). Using student-built algorithm animations as learning aids. *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE 97) (pp. 25-29), San Jose, USA: ACM.

Stasko, J., & Kraemer, E. (1993). The Visualization of Parallel Systems: An Overview, *Journal of Parallel and Distributed Computing*, *18*(2), 105-117.

Sysło, M.M. (2011) Outreach to Prospective Informatics Students. In: Kalaš I., Mittermeir R.T. (eds) *Informatics in Schools. Contributing to 21st Century Education. ISSEP 2011. Lecture Notes in Computer Science*, vol 7013. Springer, Berlin, Heidelberg.

The Royal Society. (2017). After the reboot: computing education in UK schools. Retrieved 12 June 2018, https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

Thomasson, B., Ratcliffe, M., & Thomas, L. (2006). Identifying novice difficulties in object oriented design. *SIGCSE Bulletin. 38*(3), 28-32.

Thompson, E. (2007). Holistic assessment criteria: applying SOLO to programming projects. *Proceedings of the ninth Australasian conference on Computing education - Volume 66 (ACE '07)*, Samuel Mann and Simon (Eds.), Vol. 66. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 155-162.

Thompson, E., Luxton-Reilly, A., Whalley, J., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for CS assessment. In S. Hamilton & M. Hamilton (Eds.), *Proceedings of the 10th Conference on Australasian Computing Education* (pp. 155-161). Darlinghurst: Australian Computer Society, Inc.

Thompson, P. W. (1994). Images of rate and operational understanding of the fundamental theorem of calculus. *Educational Studies in Mathematics, 26*, 229-274.

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K-12 computer science: Report of the ACM K-12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

Urquiza–Fuentes, J., & Velazquez–Iturbide, J.A. (2009). A survey of successful evaluations of program visualization and algorithm animation systems. *ACM Transactions of Computing Education, 9*(2), 1-21.

Vandenberg, S. and Wollowski, M. (2000). Introducing Computer Science Using a Breadth-First Approach and Functional Programming. *SIGCSE Bulletin, 32*(1),180–184.

Velasquez-Iturbide, J., A., Perez-Carrasco, A., & Urquiza-Fuentes, J., A. (2008). SRec: An animation system of recursion for algorithm courses. *Proceedings of the 13th Annual Conference Innovation and Technology in Computer Science Education* (pp. 225-229). Madrid: ACM.

Velazquez-Iturbide, A. J., Hernan-Losada, I. & Paredes-Velasco, M. (2017). Evaluating the effect of program visualization on student motivation. *IEEE Transactions on Education 60*(3), 238-245.

Velazquez-Iturbide, J. A. (2000). Recursion in Gradual Steps (Is recursion really that difficult?). *ACM SIGCSE Bulletin, 32*(1), 310 – 314.

Veerasamy, K. A., D'Souza, D. & Laakso M-J.(2016). Identifying novice student programming misconceptions and errors from summative assessments. *Journal of Educational Technology Systems. 45*(1). 50-73.

Vrachnos, E., & Jimoyiannis, A. (2008). Dave: A Dynamic Algorithm Visualization Environment for Novice Learners. *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies* (pp. 319-323). Santander: IEEE Computer Society.

Vrachnos, E., & Jimoyiannis, A. (2014). Design and evaluation of a web-based dynamic algorithm visualization environment for novices. *Procedia Computer Science*, 27, 229-239.

Vrachnos, E., & Jimoyiannis, A. (2017). Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy. *Themes in Science and Technology Education*, 10(1), 31-52

Watson, C. & Frederick W.B.L. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14) (pp*. 39-44*)*. New York: ACM.

Weintrop, D. & Holbert, N. (2017). From Blocks to Text and Back: Programming Patterns in a Dual-Modality Environment. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. (pp. 633-638) ACM, New York.

Weintrop, D. & Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transactions on Computing Education*, 18(1), article 3.

Whalley, J., & Kasto, N., (2014). How difficult are novice code writing tasks? A software metrics approach. In J. Whalley & D. D'Souza (Eds.), *Proceedings of the Sixteenth Australasian Computing Education Conference*, (pp. 105-112). Darlinghurst: Australian Computer Society, Inc.

Whalley, J., Clear, T., Robbins, P., & Thompson, E. (2011). Salient elements in novice solutions to code writing problems. In J. Hamer, & M. de Raadt (Eds.) *Proceedings of the Thirteenth Australasian Computing Education Conference* (pp. 37-46). Darlinghurst: Australian Computer Society, Inc.

Wilkerson-Jerde, M. H., Wagh, A. & Wilensky, U. (2015). Balancing curricular and pedagogical needs in computational construction kits: Lessons from the DeltaTick project. *Science Education, 99*(3), 465-499.

Wilson, C., Sudol, L. A., Stephenson, C., and Stehlik, M. (2010). Running on Empty. Executive Summary. Retrieved 12 June 2018 from https://runningonempty.acm.org/exec_summary.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM 49*(3), 33-35.

Winslow, L. E. (1996). Programming Pedagogy. *SIGCSE Bulletin, 28*(3), 17-22

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2014). *App Inventor 2: Create Your Own Android Apps*. 2nd ed. Beijing: O'Reilly Media.

Yamashita, K., Nagao, T., Kogure, S., Noguchi, Y., Konishi, Y. & Itoh, Y. (2016). Code-reading support environment visualizing three fields and educational practice to understand nested loops. *Research and Practice in Technology Enhanced Learning*, *11*(3).

Αναλυτικό Πρόγραμμα Σπουδών Πληροφορικής ΕΠΑΛ (ΦΕΚ 2010/Τβ/16-09-2015).

Βραχνός, Ε., & Τζιμογιάννης, Α. (2009). Εκπαιδευτικά περιβάλλοντα οπτικοποίησης αλγορίθμων: Μια επισκόπηση των τεχνικών και παιδαγωγικών χαρακτηριστικών. *Θέματα Επιστημών και Τεχνολογίας στην Εκπαίδευση*, *2*(3), 215-245.

Βραχνός, Ε., & Τζιμογιάννης, Α. (2010). Μελέτη των αναπαραστάσεων μαθητών της Γ' Λυκείου για την έννοια του πίνακα χρησιμοποιώντας την ταξινομία SOLO. *5ο Πανελλήνιο Συνέδριο "Διδακτική της Πληροφορικής"* (σ. 81-90). Αθήνα.

Βραχνός, Ε., & Τζιμογιάννης, Α. (2014α). Αναπαραστάσεις μαθητών και φοιτητών για τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής: Μια ανάλυση βασισμένη στην ταξινομία SOLO. *Πρακτικά 7ου Πανελλήνιου Συνεδρίου ''Διδακτική της Πληροφορικής''*, Ρέθυμνο.

Βραχνός, Ε., & Τζιμογιάννης, Α. (2014β). Μελέτη της συμβολής του περιβάλλοντος οπτικοποίησης αλγορίθμων DAVE στην οικοδόμηση αλγορίθμων ταξινόμησης από μαθητές Γ' Λυκείου. *Πρακτικά 7ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Ρέθυμνο.

Βραχνός, Ε., & Τζιμογιάννης, Α. (2016). Προϋπάρχουσες γνώσεις των μαθητών στους αλγόριθμους ταξινόμησης. Ποιον αλγόριθμο επινοούν; *Πρακτικά 8ου Πανελλήνιου Συνεδρίου ''Διδακτική της Πληροφορικής''*, σελ. 61-68, Ιωάννινα.

Βραχνός, Ε., & Τζιμογιάννης, Α. (2018). Προϋπάρχουσες γνώσεις των μαθητών Γυμνασίου και Λυκείου στους αλγόριθμους ταξινόμησης. Μια συγκριτική ανάλυση. *Πρακτικά 9ου Πανελλήνιου Συνεδρίου "Διδακτική της Πληροφορικής"*. Θεσσαλονίκη.

Ευρωπαϊκή εβδομάδα του Κώδικα Ανακτήθηκε 12 Ιουνίου 2018 από http://codeweek.eu/

ΥΠΔΒΜΘ (2011α). *Πρόγραμμα Σπουδών για τις ΤΠΕ στην Πρωτοβάθμια Εκπαίδευση.* Πράξη «ΝΕΟ ΣΧΟΛΕΙΟ (Σχολείο 21ου αιώνα) – Νέο πρόγραμμα σπουδών, στους Άξονες Προτεραιότητας 1, 2, 3 - Οριζόντια Πράξη», με κωδικό MIS 295450, Υποέργο 1: «Εκπόνηση Προγραμμάτων Σπουδών Πρωτοβάθμιας και Δευτεροβάθμιας Εκπαίδευσης και οδηγών για τον εκπαιδευτικό «Εργαλεία Διδακτικών Προσεγγίσεων». Αθήνα: Παιδαγωγικό Ινστιτούτο.

ΥΠΔΒΜΘ (2011β). *Πρόγραμμα Σπουδών για τις ΤΠΕ στο Γυμνάσιο..* Πράξη «ΝΕΟ ΣΧΟΛΕΙΟ (Σχολείο 21ου αιώνα) – Νέο πρόγραμμα σπουδών, στους Άξονες Προτεραιότητας 1, 2, 3 - Οριζόντια Πράξη», με κωδικό MIS 295450, Υποέργο 1: «Εκπόνηση Προγραμμάτων Σπουδών Πρωτοβάθμιας και Δευτεροβάθμιας Εκπαίδευσης και οδηγών για τον εκπαιδευτικό «Εργαλεία Διδακτικών Προσεγγίσεων». Αθήνα: Παιδαγωγικό Ινστιτούτο.

Κόμης, Β. & Μικρόπουλος, Τ. Α. (2001). *Πληροφορική στην Εκπαίδευση*. Ελληνικό Ανοικτό Πανεπιστήμιο.

Κόμης, Β., & Τζιμογιάννης, Α. (2006). Ο Προγραμματισμός ως μαθησιακή δραστηριότητα: από τις εμπειρικές προσεγγίσεις στη γνώση παιδαγωγικού περιεχομένου. *Θέματα στην Εκπαίδευση, 7*(3), 229-255.

Τζιμογιάννης, Α. (2003). Η διδασκαλία του προγραμματισμού στο Ενιαίο Λύκειο: Προς ένα ολοκληρωμένο πλαίσιο με στόχο την ανάπτυξη δεξιοτήτων επίλυσης προβλημάτων. *Πρακτικά 2ου Πανελλήνιου Συνεδρίου ''ΤΠΕ στην εκπαίδευση'* (σ.706-720). Σύρος.

Τζιμογιάννης, Α. (2005). Προς ένα παιδαγωγικό πλαίσιο διδασκαλίας του προγραμματισμού στη δευτεροβάθμια εκπαίδευση. Στο Α. Τζιμογιάννης (επιμ.), *Πρακτικά 3ου Πανελλήνιου Συνεδρίου ''Διδακτική της Πληροφορικής''* (σ. 99-111). Κόρινθος.

Τζιμογιάννης, Α. (2017). *Ηλεκτρονική Μάθηση: Θεωρητικές προσεγγίσεις και εκπαιδευτικοί σχεδιασμοί*. Αθήνα: Εκδόσεις Κριτική.

Τζιμογιάννης, Α., Τσιωτάκης, Π., & Sajaniemi, J. (2006). Μελετώντας το ρόλο των προσομοιώσεων αλγορίθμων στη διδασκαλία του προγραμματισμού στο Ενιαίο Λύκειο. Στο Ε. Σταυρίδου & Χ.

Σολομωνίδου (επιμ.), *Πρακτικά Πανελλήνιου Συνεδρίου ''Ψηφιακό Εκπαιδευτικό Υλικό: Ζητήματα δημιουργίας, διδακτικής αξιοποίησης και αξιολόγησης''* (σ. 99-108). Βόλος.

ΥΠΕΠΘ (1998). *Η Πληροφορική στο σχολείο*, Παιδαγωγικό Ινστιτούτο, Αθήνα.

Ώρα του Κώδικα, Ανακτήθηκε 12 Ιουνίου 2018 από https://hourofcode.com/

# *Παράρτημα: Ερευνητικά Εργαλεία*

## *Ερωτηματολόγιο της έρευνας για τη μελέτη των προϋπαρχουσών αντιλήψεων των μαθητών στους αλγόριθμους ταξινόμησης*

Αγαπητοί/ές μαθητές/τριες,
Το ερωτηματολόγιο αυτό στοχεύει να μελετήσει τους παράγοντες που επηρεάζουν την κατανόηση αλγορίθμων ταξινόμησης. Σας παρακαλούμε να διαβάσετε προσεκτικά τα ερωτήματα στα παρακάτω έργα-προβλήματα. Στη συνέχεια, να δώσετε τις απαντήσεις σας και να τις αιτιολογήσετε συνοπτικά.
Σας ευχαριστούμε για τη συνεργασία σας.

**ΑΤΟΜΙΚΑ ΣΤΟΙΧΕΙΑ**

**Όνομα:** .....

**Φύλο:** .....

**Τάξη :** .....

**Κατεύθυνση φοίτησης στο Λύκειο:** .....

### Έργο 1

Σας δίνουν 20 κλειστούς φακέλους. Σε κάθε έναν από αυτούς υπάρχει ένα χαρτί με έναν αριθμό γραμμένο πάνω σε αυτό. Σας ζητείται να βάλετε τους φακέλους σε αύξουσα σειρά (δηλαδή από το μικρότερο στο μεγαλύτερο) με βάση τα νούμερα αυτά.

Μπορείτε να ανοίξετε κάθε φάκελο όσες φορές θέλετε αλλά μπορείτε να έχετε ανοιχτούς την ίδια στιγμή το πολύ δύο φακέλους και να βλέπετε τι έχουν μέσα. Μετά τους κλείνετε πάλι και τους τοποθετείτε όπου εσείς κρίνετε.

Να περιγράψετε όσο πιο αναλυτικά μπορείτε τον αλγόριθμο με τον οποίο θα βάλετε τους φακέλους στη σειρά

## *Ερωτηματολόγιο της έρευνας για τη μελέτη των παρανοήσεων και των δυσκολιών των μαθητών στους πίνακες*

***Πανεπιστήμιο Πελοποννήσου***

***Τμήμα Κοινωνικής και Εκπαιδευτικής Πολιτικής***

*Αγαπητοί μαθητές,*

*Αφού διαβάσετε προσεκτικά τα ερωτήματα στα παρακάτω έργα, παρακαλούμε να δώσετε αιτιολογημένα τις απαντήσεις σας σύμφωνα με τις οδηγίες.*

*Σας ευχαριστούμε για τη συνεργασία σας.*

**Όνομα**: ..... **Επώνυμο**: .....

- Έχεις πρόσβαση σε υπολογιστή εκτός του σχολείου;

  *Σε δικό μου Η/Υ* *Σε Η/Υ φίλων* *Σε σχολές Η/Υ* *Σε Internet cafe* *Όχι δεν έχω*

- Πόσες φορές έχεις υλοποιήσει ένα πρόγραμμα σε ένα προγραμματιστικό περιβάλλον (Διερμηνευτής, Γλωσσομάθεια) στα πλαίσια του μαθήματος της Α.Ε.Π.Π;

  καμία 2–3 μέχρι 10 Πάρα πολλές

- Έχεις κάνει μαθήματα προγραμματισμού στο γυμνάσιο; Αν ναι ποια γλώσσα προγραμματισμού χρησιμοποιήσατε;

  ΓΛΩΣΣΑ Pascal Basic Logo Άλλη .....

- Έχεις κάνει μαθήματα προγραμματισμού στην Α' Λυκείου ; Αν ναι ποια γλώσσα προγραμματισμού χρησιμοποιήσατε;

  ΓΛΩΣΣΑ Pascal Basic Logo Άλλη .....

- Πόσο συχνά χρησιμοποιείτε το εργαστήριο Πληροφορικής για εξάσκηση στον προγραμματισμό (στο μάθημα ΑΕΠΠ);

  1 ώρα την εβδομάδα 1 ώρα κάθε 15 μέρες 1 ώρα κάθε μήνα σπανιότερα καθόλου

- Για εσένα ο προγραμματισμός είναι ένα ιδιαίτερα ενδιαφέρον αντικείμενο

  Διαφωνώ απόλυτα Διαφωνώ Ουδέτερη άποψη Συμφωνώ Συμφωνώ απόλυτα

- Θεωρείς τον προγραμματισμό δύσκολο;

  Διαφωνώ απόλυτα Διαφωνώ Ουδέτερη άποψη Συμφωνώ Συμφωνώ απόλυτα

## Ζήτημα 1

Έστω οι ακέραιες μεταβλητές **X**, **Y** και ο πίνακας ακεραίων **A**[10]. Να γραφούν σε ΓΛΩΣΣΑ οι εντολές που υλοποιούν τις παρακάτω λειτουργίες:

**i)** Εκχώρηση του αριθμού 3 στο 4ο στοιχείο του πίνακα **A** **……....**
**ii)** Εκχώρηση της τιμής του 5ου στοιχείου του πίνακα **A** στο 7ο στοιχείο του πίνακα **A** **…….....**
**iii)** Εμφάνιση στην οθόνη της θέσης του στοιχείου **A**[X] του πίνακα **A** **……..**
**iv)** Αύξηση του 5ου στοιχείου του πίνακα **A** κατά 3 **…….....**

## Ζήτημα 2

Έστω ότι **Π**, **Β** είναι πίνακες ακέραιων αριθμών 10 στοιχείων ο κάθε ένας. Τι αναμένεις να εμφανιστεί στην οθόνη μετά την εκτέλεση των παρακάτω εντολών; Να αιτιολογήσεις την απάντησή σου

| **Αλγόριθμος** | **Απάντηση** |
|---|---|
| α ← 1<br>Π[ α ] ← 2<br>Π[ Π[α] ] ← 3<br>Β[1] ← 2<br>Β[Π[2]] ← 10<br>**Γράψε** Β[3], Π[2] | |

## Ζήτημα 3

Έστω ότι σε ένα πρόγραμμα υπάρχει η παρακάτω δήλωση

**ΜΕΤΑΒΛΗΤΕΣ**

**ΑΚΕΡΑΙΕΣ**: i[10], A, j

Τι αναμένεις να εμφανιστεί κατά την εκτέλεση των παρακάτω εντολών;

| **Αλγόριθμος** | **Απάντηση** |
|---|---|
| A ← 1<br>j ← A + 1<br>i[A+1] ← 10<br>i[A] ← i[A+1] + i[j]<br>**Γράψε** i[A], i[j] | |

**Ζήτημα 4**

Δίνεται ο πίνακας ακέραιων αριθμών **A,** με τιμές όπως παρακάτω

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 3 | 5 | 7 | 8 |

Έστω ότι ένας αλγόριθμος περιέχει τις ακόλουθες εντολές. Να εξηγήσεις τι κάνει κάθε εντολή. (Σημείωσε στο πλαίσιο δεξιά από κάθε εντολή)

| | |
|---|---|
| x ← 2 | Εκχώρηση της τιμής 2 στη μεταβλητή x |
| A[3] ← 5 | |
| A[5] ← 5 | |
| A[x] ← A[x] + 1 | |
| A[x] ← A[x] + A[1] | |
| A[x+1] ← A[x] | |
| A[x+1] ← A[x+1] + 1 | |

## Ζήτημα 5

Δίνονται δύο τμήματα αλγορίθμων τα οποία διαβάζουν 100 βαθμούς μαθητών με αποδεκτές τιμές στο διάστημα [1,20] και τους καταχωρούν σε ένα πίνακα.

| **Αλγόριθμος 1** | **Αλγόριθμος 2** |
|---|---|
| **Για** i **από** 1 **μέχρι** N<br>**Αρχή_Επανάληψης**<br>**Διάβασε** βαθμός<br>**Μέχρις_ότου** βαθμός >= 1 **και** βαθμός <=20<br>B[i] ← βαθμός<br>**Τέλος_Επανάληψης** | **Για** i **από** 1 **μέχρι** N<br>**Αρχή_Επανάληψης**<br>**Διάβασε** B[i]<br>**Μέχρις_ότου** B[i] >= 1 **και** B[i] <=20<br>**Τέλος_Επανάληψης** |

Ποιο από τα παραπάνω τμήματα αλγορίθμου θα επέλεγες να γράψεις εσύ; Να αιτιολογήσεις την απόφασή σου.

## Ζήτημα 6

Ο παρακάτω αλγόριθμος υπολογίζει τον μεγαλύτερο από τους βαθμούς 50 μαθητών μιας τάξης. Να τροποποιήσεις τον αλγόριθμο έτσι ώστε να υπολογίζει και να εμφανίζει πόσοι μαθητές έχουν τον μεγαλύτερο βαθμό.

(*Στη γενική περίπτωση, μπορεί να υπάρχουν περισσότεροι του ενός μαθητές που έχουν τον μεγαλύτερο βαθμό*)

| | |
|---|---|
| **Διάβασε** βαθ<br>Max ← βαθ<br>**Για** μαθητή **Από** 2 **Μέχρι** 50<br>  **Διάβασε** βαθ<br>  **Αν** βαθ > max **Τότε**<br>  max ← βαθ<br>  **Τέλος_Αν**<br>**Τέλος_Επανάληψης** | |

## Ζήτημα 7

Δίνεται ο πίνακας ακέραιων αριθμών **X** με τιμές όπως παρακάτω

| 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|

Τι περιμένετε να εμφανιστεί στην οθόνη κατά την εκτέλεση του παρακάτω προγράμματος; Ποιες θα είναι οι τελικές τιμές των στοιχείων του πίνακα; Να αιτιολογήσετε την απάντησή σας.

```
i ← 1
Όσο i < X[i] Επανάλαβε
    X[X[i]] ← X[i]
    i ← i + 1
   Εμφάνισε i, X[i], X[X[i]]
Τέλος_Επανάληψης
```

## *Ερωτηματολόγιο της έρευνας για τη μελέτη των δυσκολιών των μαθητών στον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής*

Αγαπητοί/ές μαθητές/τριες,
Το ερωτηματολόγιο αυτό στοχεύει να μελετήσει τους παράγοντες που επηρεάζουν την κατανόηση αλγορίθμων ταξινόμησης σε μονοδιάστατους πίνακες. Σας παρακαλούμε να διαβάσετε προσεκτικά τα ερωτήματα στα παρακάτω έργα-προβλήματα. Στη συνέχεια, να δώσετε τις απαντήσεις σας και να τις αιτιολογήσετε συνοπτικά.

Σας ευχαριστούμε για τη συνεργασία σας.

**ΑΤΟΜΙΚΑ ΣΤΟΙΧΕΙΑ**

**Φύλο:** ..... ☐

**Ηλικία:** ..... ☐

**Έργο 1**

Έστω ο πίνακας **Α** με στοιχεία όπως παρακάτω:

| | | | | | |
|---|---|---|---|---|---|
| 40 | 35 | 25 | 15 | 10 | 5 |

Στον πίνακα **Α** εκτελείται ο αλγόριθμος ταξινόμησης της φυσαλίδας. Θεωρείστε ότι, για κάποιον απρόβλεπτο (τεχνικό) λόγο, ο αλγόριθμος σταματάει χωρίς να έχει ολοκληρωθεί η ταξινόμηση. Έστω ότι τη στιγμή της διακοπής ο πίνακας έχει τη μορφή:

| | | | | | |
|---|---|---|---|---|---|
| 5 | 10 | 15 | 40 | 25 | 35 |

**α)** Η ταξινόμηση είναι αύξουσα ή φθίνουσα; Να αιτιολογήσετε την απάντησή σας.

**β)** Πόσα στοιχεία του πίνακα έχουν ήδη ταξινομηθεί (δηλαδή βρίσκονται στις τελικές θέσεις ταξινόμησης σύμφωνα με τον αλγόριθμο φυσαλίδας);

**γ)** Ποια είναι τα επόμενα στοιχεία που θα συγκριθούν, όταν η ταξινόμηση ξεκινήσει ξανά από το σημείο διακοπής;

**δ)** Ποια είναι τα επόμενα στοιχεία που θα αλλάξουν θέση, όταν η ταξινόμηση ξεκινήσει ξανά από το σημείο διακοπής;

**Έργο 2**

Δίνεται ο πίνακας Α με Ν ακέραιους αριθμούς και ο παρακάτω αλγόριθμος

```
Για i από N μέχρι 2 με βήμα –1
    Αν A[i] < A[i–1] Τότε
        temp ← A[i]
        A[i]  ← A[i–1]
        A[i–1] ← temp
    Τέλος_Αν
Τέλος_Επανάληψης
```

Τι αναμένετε να συμβεί κατά την εκτέλεση του αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του.

**Έργο 3**

Δίνεται ο πίνακας **Α** με **Ν** ακέραιους αριθμούς και ο παρακάτω αλγόριθμος:

```
Για i από N–1  μέχρι 1 με βήμα –1
   Για j από 1 μέχρι i
       Αν A[j] < A[j+1] Τότε
           temp ← A[j]
           A[j]  ← A[j+1]
           A[j+1] ← temp
       Τέλος_Αν
   Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

Τι αναμένετε να συμβεί κατά την εκτέλεση του αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του.

**Έργο 4**

Έστω ο πίνακας **Α** με στοιχεία όπως παρακάτω:

| 39 | 23 | 29 | 14 | 38 | 23 |
|---|---|---|---|---|---|

Ποια θα είναι η μορφή του πίνακα Α, μετά την εκτέλεση του παρακάτω αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του αλγορίθμου.

```
Για i από 4 μέχρι 6
   Για j από 6 μέχρι i με βήμα –1
      Αν A[j] < A[j–1] Τότε
         temp ← A[j]
         A[j]  ← A[j–1]
         A[j–1] ← temp
      Τέλος_Αν
   Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |

## Έργο 5

Δίνεται ο παρακάτω αλγόριθμος και οι πίνακες **Όνομα** και **Βαθμός** που αφορούν τα ονόματα και τις βαθμολογίες 5 μαθητών.

```
Για i από 2 μέχρι 5
   Για j από 5 μέχρι i με βήμα –1
      Αν Βαθμός[j] > Βαθμός[j–1] Τότε
            Αντιμετάθεσε  Βαθμός[j], Βαθμός[j–1]
            Αντιμετάθεσε  Όνομα[j], Όνομα[j–1]
      Τέλος_Αν
   Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

| | Όνομα | Βαθμός |
|---|---|---|
| 1 | Θανάσης | 18 |
| 2 | Μυρτώ | 19 |
| 3 | Ελένη | 18 |
| 4 | Μαρία | 20 |
| 5 | Γεωργία | 18 |

**α)** Ποια θα είναι τα πρώτα τέσσερα ονόματα στον πίνακα **Όνομα** μετά την εκτέλεση του αλγορίθμου;

**β)** Τί αναμένετε να συμβεί στους 2 πίνακες, μετά την εκτέλεση του νέου αλγορίθμου, αν η συνθήκη **Βαθμός[j] > Βαθμός[j–1]** αλλάξει σε **Βαθμός[j] >= Βαθμός[j–1]**. Να αιτιολογήσετε την απάντησή σας.

**Έργο 6**

Έστω ο πίνακας **Α** με στοιχεία όπως παρακάτω:

| 15 | 10 | 15 | 12 | 1 | 8 |
|---|---|---|---|---|---|

Ποια θα είναι η μορφή του πίνακα Α, μετά την εκτέλεση του παρακάτω αλγορίθμου; Να εξηγήσετε συνοπτικά τη λειτουργία του αλγορίθμου.

```
Για i από 2 μέχρι 6
   Για j από 6 μέχρι i με βήμα –1
      Αν  A[j] mod 2 < A[j–1] mod 2  Τότε
         temp ← A[j]
         A[j] ← A[j–1]
         A[j–1] ← temp
      Τέλος_Αν
   Τέλος_Επανάληψης
Τέλος_Επανάληψης
```

*Υπενθυμίζεται ότι η έκφραση **α mod 2**, υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του αριθμού α με το 2. Για παράδειγμα, 4 mod 2 = 0, 3 mod 2 = 1, 1 mod 2 = 1.*

| | | | | | |
|---|---|---|---|---|---|

## *Ερωτηματολόγιο της έρευνας για τη μελέτη της συμβολής του λογισμικού οπτικοποίησης αλγορίθμων DAVE*

### Φύλλο Εργασίας Μαθητή

**Ενότητα:** *Ταξινόμηση Ευθείας Ανταλλαγής*
**Χώρος :** *Εργαστήριο Η/Υ* **Χρόνος:** *2 διδακτικές ώρες*

***Ονοματεπώνυμο:*** _____

***Τμήμα:*** _____

**Σκοπός**

Η δραστηριότητα αυτή αφορά στη μελέτη του αλγορίθμου ταξινόμησης της ευθείας ανταλλαγής με χρήση του περιβάλλοντος δυναμικής οπτικοποίησης αλγορίθμων DAVE.

**Οδηγίες για το περιβάλλον οπτικοποίησης**

1. Στις ερωτήσεις, όπου ζητείται να τροποποιήσετε τον αλγόριθμο, να μεταφέρετε στο φύλλο εργασίας μόνο τις εντολές που τροποποιήσατε και όχι ολόκληρο τον αλγόριθμο
2. Στον αλγόριθμο που δίνεται, **Ν είναι το πλήθος των στοιχείων του πίνακα**.
3. Αν σε οποιοδήποτε σημείο το περιβάλλον οπτικοποίησης παρουσιάσει πρόβλημα ή θέλετε να μεταβείτε στην αρχική κατάσταση, **πατήστε το πλήκτρο F5 για να ξεκινήσει η εφαρμογή από την αρχή**.
4. Στο περιβάλλον οπτικοποίησης μπορείτε να τροποποιήσετε τα παρακάτω στοιχεία του αλγορίθμου, όπως φαίνεται και στον επόμενο πίνακα
   - Τις αρχικές, τελικές τιμές και το βήμα κάθε επανάληψης **(**κενά **1 – 6)**
   - Τον συγκριτικό τελεστή της συνθήκης στη δομή επιλογής **(**κενό **7)**
   - Τις αριθμητικές εκφράσεις που βρίσκονται μέσα στις αγκύλες, π.χ. μπορείτε να τροποποιήσετε την έκφραση A[j-1] σε A[j+1] ή A[2*j+1] ή A[j-2] (κενά **8 – 11 )**

Παρακάτω δίνεται ένα παράδειγμα με τα σημεία του αλγορίθμου της φυσαλίδας που μπορείτε να τροποποιήσετε (1 – 11).

| **Αλγόριθμος στο περιβάλλον DAVE αλλαγών** | **Παράδειγμα** |
|---|---|
| Για i από **..( 1 )..** μέχρι **..( 2 )..** με βήμα **..( 3 )..**<br>Για j από **..( 4 )..** μέχρι **..( 5 )..** με βήμα **..( 6 )..**<br>Αν A[**..( 8 )..** ] **..( 7 )..** A[**..( 9 )..** ] Τότε<br>Αντιμετάθεσε A[**..( 10 )..** ], A[**..( 11 )..** ]<br>Τέλος_Αν<br>Τέλος_Επανάληψης<br>Τέλος_Επανάληψης | Για i από **2** μέχρι **2** με βήμα **1**<br>Για j από **N** μέχρι **i** με βήμα **–1**<br>Αν A[**j**] **<** A[**j–1**] Τότε<br>Αντιμετάθεσε A[**j**], A[**j – 1**]<br>Τέλος_Αν<br>Τέλος_Επανάληψης<br>Τέλος_Επανάληψης |

# Δραστηριότητες Φύλλου Εργασίας

**Δραστηριότητα 1**

**Βήμα 1**

Ανοίξτε έναν φυλλομετρητή (κατά προτίμηση τον Chrome) μεταβείτε στην ιστοσελίδα http://evripides.mysch.gr/dave και επιλέξτε "*Ταξινόμηση Ευθείας Ανταλλαγής*"

**Βήμα 2**

Να εκτελέστε τον αλγόριθμο όπως δίνεται, επιλέγοντας ***Βήμα – Βήμα*** .

1. Παρατηρήστε την οπτικοποίηση του αλγορίθμου και εντοπίστε την εντολή που εκτελείται κάθε φορά και την αντίστοιχη οπτικοποίησή της.
2. Περιγράψτε συνοπτικά τη λειτουργία του αλγορίθμου
3. Ποια είναι τα δύο πρώτα στοιχεία που θα αντιμετατεθούν; Γιατί;
4. Ποιο είναι το τελικό αποτέλεσμα που αναμένεις να του αλγορίθμου;

**Βήμα 3**

Να εκτελέσετε τον αλγόριθμο ακόμη δύο φορές. Τι παρατηρείτε; Τι θα συμβεί αν ο αλγόριθμος εκτελεστεί πολλές φορές; Να αιτιολογήσετε την απάντησή σας.

**Βήμα 4**

Να τροποποιήσετε τον αλγόριθμο έτσι ώστε στην πρώτη θέση του πίνακα να μεταφερθεί ο μεγαλύτερος αριθμός. Να αιτιολογήσετε την απάντησή σας.

**Βήμα 5**
Να τροποποιήσετε τον αλγόριθμο έτσι ώστε στις 3 πρώτες θέσεις να βρεθούν οι 3 μεγαλύτεροι αριθμοί. Να αιτιολογήσετε την απάντησή σας.

**Βήμα 6**
Τι πρέπει να αλλάξετε στον αλγόριθμο έτσι ώστε να ταξινομηθούν όλα τα στοιχεία του πίνακα σε φθίνουσα σειρά;

**Δραστηριότητα 2**
Στο πεδίο "*Εισαγωγή Προκαθορισμένου Πίνακα*" δώστε τους παρακάτω 6 αριθμούς χωρισμένους με ένα κενό: 10 10 10 10 10 3 και πατήστε **Εισαγωγή**.
Στη συνέχεια τροποποιήστε τον αλγόριθμο έτσι ώστε να εκτελεί αύξουσα ταξινόμηση. Πόσα περάσματα θα χρειαστούν για να ταξινομηθεί ο πίνακας; Να αιτιολογήσετε την απάντησή σας.

**Δραστηριότητα 3**
Εισάγετε πίνακα με **τα εξής πέντε στοιχεία:** 1 4 3 2 5. Πόσα περάσματα θα χρειαστούν ώστε να ταξινομηθεί (με ποια σειρά εννοείς;) ο πίνακας αυτός; Πόσες αντιμεταθέσεις θα γίνουν; Μπορείτε να προτείνετε έναν τρόπο ταξινόμησης του πίνακα με πολύ λιγότερες αντιμεταθέσεις; Να αιτιολογήσετε την απάντησή σας.

**Δραστηριότητα 4**

Εισάγετε τα 8 στοιχεία του πίνακα *1 0 1 0 1 0 1 0*. Πόσα περάσματα θα χρειαστούν για να ταξινομηθεί ο πίνακας σε αύξουσα σειρά; Πόσες αντιμεταθέσεις θα γίνουν; Τροποποιήστε τον αλγόριθμο αλλάζοντας τον συγκριτικό τελεστή από **A[j] < A[j–1]** σε **A[j] <= A[j–1]** . Εκτελέστε τον αλγόριθμο. Τι παρατηρείτε; Μετά από πόσα περάσματα θα σταματήσουν οι αντιμεταθέσεις;

**Δραστηριότητα 5**

Εισάγετε τον πίνακα 7 στοιχείων 32 38 98 54 60 90 20

Να σχεδιάσετε τον αλγόριθμο ταξινόμησης της φυσαλίδας για την περίπτωση που η φυσαλίδα κατευθύνεται προς τα κάτω και όχι προς τα πάνω σε έναν πίνακα **A**, **N** θέσεων. Δηλαδή ξεκινάει από τη θέση 1 και κινείται προς το τέλος του πίνακα όπως φαίνεται παρακάτω για έναν πίνακα 7 θέσεων.

Σημειώστε τον τρόπο σκέψης σας και τις δοκιμές ή τα λάθη που έγιναν μέχρι να καταλήξετε στον ζητούμενο αλγόριθμο.

| | **1ο πέρασμα** | | | | | |
|---|---|---|---|---|---|---|
| 1 | 32 | 38 | 38 | 38 | 38 | 38 |
| 2 | 38 | 32 | 98 | 98 | 98 | 98 |
| 3 | 98 | 98 | 32 | 54 | 54 | 54 |
| 4 | 54 | 54 | 54 | 32 | 60 | 60 |
| 5 | 60 | 60 | 60 | 60 | 32 | 90 |
| 6 | 90 | 90 | 90 | 90 | 90 | **32** |
| 7 | **20** | **20** | **20** | **20** | **20** | **20** |

| **2ο πέρασμα** | | | | | |
|---|---|---|---|---|---|
| 38 | 98 | 98 | 98 | 98 | 98 |
| 98 | 38 | 54 | 54 | 54 | 54 |
| 54 | 54 | 38 | 60 | 60 | 60 |
| 60 | 60 | 60 | 38 | 90 | 90 |
| 90 | 90 | 90 | 90 | **38** | **38** |
| **32** | **32** | **32** | **32** | **32** | **32** |
| **20** | **20** | **20** | **20** | **20** | **20** |

**Εξώφυλλο:**

Η έγκριση της διδακτορικής διατριβής από το τμήμα Κοινωνικής και Εκπαιδευτικής Πολιτικής της Σχολής Κοινωνικών και Πολιτικών Επιστημών του Πανεπιστήμιου Πελοποννήσου δεν υποδηλώνει αποδοχή των απόψεων του συγγραφέα (σύμφωνα με τις διατάξεις του άρθρου 201, παράγραφος 2 του ν. 5343/1932).