Η εισαγωγή στον προγραμματισμό: Διδακτικές Προσεγγίσεις και Εκπαιδευτικά Εργαλεία

Σ. Ξυνόγαλος[(1)], Μ. Σατρατζέμη[(2)], Β. Δαγδιλέλης[(3)]

(1) Υποψήφιος Διδάκτωρ, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας

(2) Επίκουρος Καθηγήτρια, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας

(3) Λέκτορας, Παιδαγωγική Σχολή Φλώρινας, ΑΠΘ

{stelios, maya, dagdil}@macedonia.uom.gr

Περίληψη

Στην εργασία αυτή δίνεται μια συνθετική παρουσίαση των χαρακτηριστικών των διάφορων *προσεγγίσεων διδασκαλίας του προγραμματισμού* σε αρχάριους και των πιο αντιπροσωπευτικών *εκπαιδευτικών εργαλείων* που έχουν αναπτυχθεί στα πλαίσια της κάθε προσέγγισης, με στόχο τη στήριξη ενός εισαγωγικού μαθήματος στον προγραμματισμό.

Λέξεις κλειδιά: εισαγωγή στον προγραμματισμό, διδακτικές προσεγγίσεις, εκπαιδευτικά εργαλεία, μικρόκοσμοι - μικρογλώσσες προγραμματισμού, εικονικές γλώσσες προγραμματισμού, συστήματα δυναμικής προσομοίωσης εκτέλεσης προγραμμάτων, δυναμική ηχητική προσομοίωση της εκτέλεσης προγραμμάτων.

Abstract

In this paper we review the various *approaches to teaching programming to novices* and the most representative *educational tools* that have been developed in the context of each approach, with the goal of supporting an introductory programming course.

1. Εισαγωγή

Η διδασκαλία και εκμάθηση του προγραμματισμού, όπως είναι γνωστό, παρουσιάζει αρκετές δυσκολίες. Σε πολλά Πανεπιστήμια του εξωτερικού, τις τελευταίες δεκαετίες, και πιο πρόσφατα και της Ελλάδας, έχει γίνει αξιόλογη ερευνητική δουλειά στην περιοχή της *Διδακτικής της Πληροφορικής* και ειδικότερα στην περιοχή της διδασκαλίας του προγραμματισμού. Όπως προκύπτει από τη σχετική βιβλιογραφία ([25]), ένας από τους σημαντικότερους παράγοντες που έχει διαπιστωθεί ότι αποτελεί πηγή δυσκολιών για την εκμάθηση του προγραμματισμού έγκειται στο γεγονός ότι, όπως φαίνεται, η κλασική προσέγγιση διδασκαλίας (classic approach [5]) είναι ασύμβατη με τις πραγματικές διδακτικές ανάγκες των μαθητών. Με τον όρο κλασική προσέγγιση διδασκαλίας εννοούμε τη διδασκαλία που συνίσταται:

- στη χρήση μιας γλώσσας γενικού σκοπού (όπως οι Pascal, C, κλπ),
- ενός επαγγελματικού περιβάλλοντος προγραμματισμού για τη γλώσσα αυτή, και
- στην επίλυση ενός συνόλου προβλημάτων επεξεργασίας αριθμών και συμβόλων.

Στην παρούσα εργασία παρουσιάζονται συνοπτικά τα χαρακτηριστικά της κλασικής προσέγγισης διδασκαλίας του προγραμματισμού και των κυριότερων *διδακτικών προβλημάτων* που εμφανίζονται στα πλαίσια της. Στη συνέχεια δίνεται μια σύντομη περιγραφή των χαρακτηριστικών και των πλεονεκτημάτων των εναλλακτικών προσεγγίσεων διδασκαλίας του προγραμματισμού που αναπτύχθηκαν έχοντας ως στόχο τη στήριξη της διδακτικής πράξης και την εξάλειψη των σχετικών διδακτικών προβλημάτων. Τέλος παρουσιάζονται μερικά από τα πιο αντιπροσωπευτικά εκπαιδευτικά εργαλεία που έχουν αναπτυχθεί στα πλαίσια της κάθε προσέγγισης.

Στόχος αυτής της ταξινόμησης των διδακτικών προσεγγίσεων και εκπαιδευτικών εργαλείων είναι να καλύψει το κενό που υπάρχει στη βιβλιογραφία για το εν λόγω θέμα. Ιδιαίτερη προσοχή δόθηκε στο να παρουσιαστούν και να σχολιαστούν σημαντικές μέθοδοι, πηγές και

προτάσεις που μπορούν να προσαρμοστούν και να χρησιμοποιηθούν στη διδακτική πράξη και να αποτελέσουν οδηγό για τους ερευνητές που έχουν ως αντικείμενο έρευνας τη διδασκαλία και εκμάθηση του προγραμματισμού.

2. Η Κλασική Προσέγγιση Διδασκαλίας του Προγραμματισμού

Περιγραφή. Οπως ήδη αναφέρθηκε και στην εισαγωγή η κλασική προσέγγιση διδασκαλίας του προγραμματισμού συνίσταται:

- στη χρήση μιας γλώσσας γενικού σκοπού (όπως οι Pascal, C, κλπ)
- ενός επαγγελματικού περιβάλλοντος προγραμματισμού για τη γλώσσα αυτή, και
- την επίλυση ενός συνόλου προβλημάτων επεξεργασίας αριθμών και συμβόλων.

Μειονεκτήματα. Σύμφωνα με αρκετούς ερευνητές, η υιοθέτηση της κλασικής προσέγγισης διδασκαλίας του προγραμματισμού αποτελεί έναν από τους σημαντικότερους παράγοντες που καθιστά την εκμάθηση του προγραμματισμού δύσκολη. Τα σημαντικότερα προβλήματα που αντιμετωπίζουν οι αρχάριοι προγραμματιστές που διδάσκονται τις αρχές του προγραμματισμού με την κλασσική προσέγγιση, όπως προκύπτει από εμπειρικές παρατηρήσεις και μελέτες ερευνητών αλλά και μαρτυρίες των ίδιων των σπουδαστών, είναι τα εξής:

1. Οι γλώσσες προγραμματισμού γενικού σκοπού διαθέτουν κατά κανόνα ένα μεγάλο ρεπερτόριο εντολών και είναι πολύπλοκες ([5], [17]).

2. Η προσοχή των μαθητών επικεντρώνεται στην εκμάθηση της σύνταξης της γλώσσας και όχι στην ανάπτυξη ικανοτήτων επίλυσης προβλημάτων ([5], [10], [23]).

3. Δεν υπάρχει, κατά κανόνα, επαρκής στήριξη του σπουδαστή στην κατανόηση των βασικών ενεργειών και δομών ελέγχου, αφού το περιβάλλον προγραμματισμού συνήθως δεν παρέχει δυνατότητες οπτικοποίησης (η διαδικασία εκτέλεσης ενός προγράμματος δεν είναι ορατή από τον σπουδαστή) ([5], [18]).

4. Οι εμπορικοί μεταγλωττιστές δεν ικανοποιούν τις ανάγκες των αρχάριων προγραμματιστών ([10], [20]).

5. Η διανοητική πολυπλοκότητα που απαιτεί η εκφορά ενός αλγορίθμου σε μια γλώσσα προγραμματισμού είναι μεγάλη, λόγω της «φύσης» της γλώσσας ([21], [14]).

6. Η επίλυση ενδιαφερόντων προβλημάτων απαιτεί την εκμάθηση ενός μεγάλου υποσυνόλου της γλώσσας και την ανάπτυξη αρκετά μεγάλων προγραμμάτων, απαιτεί δηλαδή τη επικέντρωση της προσοχής στην εκμάθηση της γλώσσας (βλέπε παραπάνω σημείο 2) [5].

3. Εναλλακτικές Προσεγγίσεις Διδασκαλίας του Προγραμματισμού

3.1 Μικρόκοσμοι – Μικρογλώσσες Προγραμματισμού

Περιγραφή. Η βασική ιδέα των *μικρόκοσμων* (microworlds) και των *μικρογλωσσών* (mini-languages) προγραμματισμού είναι η δημιουργία μιας μικρής και απλής γλώσσας προγραμματισμού για τη στήριξη των πρώτων βημάτων της εκμάθησης του προγραμματισμού. Η πλειοψηφία των μικρόκοσμων προγραμματισμού ενσωματώνει ένα ανοιχτό περιβάλλον που βασίζεται σε κάποιο φυσικό μοντέλο, ενώ ο χρήστης ελέγχει ένα *πρωταγωνιστή* που «ζει» στο περιβάλλον αυτό. Ο πρωταγωνιστής μπορεί να είναι μια χελώνα, ένα ρομπότ ή κάποια άλλη οντότητα. Ο σπουδαστής μαθαίνει βασικές έννοιες του προγραμματισμού ελέγχοντας με τις διαθέσιμες εντολές τον πρωταγωνιστή του μικρόκοσμου. Οι εντολές είναι ιδιαίτερα απλοποιημένες, ενώ το αποτέλεσμα της εκτέλεσης τους είναι ορατό στην οθόνη. Συγκεκριμένα, ο χρήστης βλέπει τον πρωταγωνιστή του μικρόκοσμου να εκτελεί μία προς μία τις εντολές, οι οποίες μεταβάλλουν την κατάστασή του και την κατάσταση του περιβάλλοντος στο οποίο ζει. Αν και το σύνολο των διαθέσιμων εντολών είναι μικρό, ωστόσο υπάρχει η δυνατότητα επίλυσης τόσο απλών όσο και αρκετά πολύπλοκων προβλημάτων. Εκτός από τις εντολές ελέγχου τις οποίες εκτελεί ο πρωταγωνιστής του μικρόκοσμου υπάρχουν και αρκετά ερωτήματα (συναρτήσεις) στα οποία μπορεί να απαντήσει, έχει δηλαδή τη δυνατότητα να ελέγχει την κατάσταση του κόσμου που «ζει». Επίσης, οι περισσότεροι μικρόκοσμοι περιλαμβάνουν όλες τις βασικές δομές ελέγχου και ένα μηχανισμό δημιουργίας νέων εντολών και υποπρογραμμάτων.

Πλεονεκτήματα. Οι μικρόκοσμοι μπορούν να χρησιμοποιηθούν αποτελεσματικά στα πλαίσια εκμάθησης εννοιών του προγραμματισμού ή και της επιστήμης των υπολογιστών γενικότερα ή/και την απόκτηση ικανοτήτων επίλυσης προβλημάτων και αλγοριθμικού τρόπου σκέψης. Η διδασκαλία του προγραμματισμού με την προσέγγιση των μικρόκοσμων παρουσιάζει αρκετά πλεονεκτήματα:

- Η γλώσσα προγραμματισμού αποτελείται από ένα περιορισμένο ρεπερτόριο εντολών με απλή σύνταξη και σημασιολογία.
- Βασίζονται σε υπαρκτά μοντέλα που είναι ήδη γνωστά στο σπουδαστή, μειώνοντας έτσι δραματικά τη διανοητική «απόσταση» ανάμεσα στα νοητά μοντέλα ή την περιγραφή σε φυσική γλώσσα των αλγορίθμων και στην περιγραφή τους στη γλώσσα προγραμματισμού.
- Τα προβλήματα που καλούνται να λύσουν οι σπουδαστές παρουσιάζουν ιδιαίτερο ενδιαφέρον.
- Η εκτέλεση ενός προγράμματος είναι ορατή, αποκαλύπτοντας έτσι τη σημασία των διδασκόμενων δομών, καθώς και τις έννοιες που σχετίζονται με τη δομή και την εκτέλεση των προγραμμάτων.
- Υπάρχει δυνατότητα προσαρμογής του μικρόκοσμου στις ανάγκες του κοινού στο οποίο απευθύνεται.

Εκπαιδευτικά εργαλεία. Το πιο γνωστό παράδειγμα μικρόκοσμου είναι η Logo, η οποία σχεδιάστηκε από τον Papert. Αν και δεν σχεδιάστηκε ειδικά για τη διδασκαλία του προγραμματισμού, αποτέλεσε ένα χρήσιμο εργαλείο για τη στήριξή της, με αποτέλεσμα να θεωρείται το πρώτο παράδειγμα μικρόκοσμου. Ωστόσο, ένας από τους πιο δημοφιλείς μικρόκοσμους προγραμματισμού όπως επισημαίνουν αρκετοί ερευνητές (Brusilovsky, Calabrese, Hvorenky, Miller, Kouchnirenko) είναι ο "***Karel the Robot***" [15] που σχεδιάστηκε από τον Richard E. Pattis προκειμένου να χρησιμοποιηθεί για το μάθημα της εισαγωγής στον προγραμματισμό. Ο Karel, ο πρωταγωνιστής του μικρόκοσμου, εκτελεί διάφορες αποστολές (προγράμματα) σε ένα κόσμο που αποτελείται από οριζόντιους δρόμους και κάθετες λεωφόρους. Στον κόσμο του Karel μπορεί να υπάρχουν τμήματα τοίχου που τοποθετούνται μεταξύ των διασταυρώσεων δημιουργώντας εμπόδια (π.χ. λαβύρινθους) που καλείται να ξεπεράσει ο Karel και beepers, μικροί πλαστικοί κώνοι που παράγουν ένα ήχο (μπιπ). Ο Karel έχει τη δυνατότητα να κινείται προς την τρέχουσα κατεύθυνση κατά 1 μπλοκ, να στρίβει κατά 90 μοίρες, να εντοπίζει beepers που βρίσκονται στην ίδια διασταύρωση μ' αυτόν, να εντοπίζει τοίχους που βρίσκονται μπροστά του σε απόσταση μισού μπλοκ χρησιμοποιώντας μια κάμερα, να σηκώνει και να κατεβάζει beepers με το μηχανικό του χέρι και τέλος μπορεί να καθορίζει προς ποια κατεύθυνση βλέπει χρησιμοποιώντας την πυξίδα του. Η γλώσσα προγραμματισμού εκτός από τις εντολές ελέγχου του Karel (move, turnleft, pickbeeper, putbeeper) περιλαμβάνει όλες τις βασικές δομές ελέγχου (if, if-else, while, loop).

Η καλύτερη υλοποίηση του "Karel the Robot", και πιθανότατα ο καλύτερος μικρόκοσμος προγραμματισμού που υπάρχει, είναι το λογισμικό ***Karel Genie*** [11] που αναπτύχθηκε στα πλαίσια του project MacGnome στο Carnegie Mellon University. Το Karel Genie βασίζεται στην τεχνολογία των *εκδοτών δομής* (structure editors) που αποτελεί αντικείμενο μελέτης στο Carnegie Mellon University από τις αρχές τις δεκαετίας του 1980. Ο εκδότης δομής παρέχει ένα μενού με τις συντακτικά σωστές μετατροπές / προσθήκες που μπορούν να γίνουν για κάθε ημιτελές τμήμα ενός προγράμματος, με αποτέλεσμα ο χρήστης να μην πληκτρολογεί εντολές, παρά μόνο κάποια αναγνωριστικά (ονόματα μεταβλητών, νέων εντολών κ.τ.λ). Επίσης, σε αντίθεση με τα συμβατικά περιβάλλοντα ο εκδότης δομής παράγει το συντακτικό δένδρο ενώ αναπτύσσεται το πρόγραμμα, με αποτέλεσμα τα προγράμματα που αναπτύσσονται να είναι πάντα συντακτικά σωστά. Επιπλέον, το Karel Genie παρέχει *πολλαπλές απόψεις* (multiple views) ενός προγράμματος ταυτόχρονα. Συγκεκριμένα, ο χρήστης έχει τη δυνατότητα να δει σε διαφορετικά παράθυρα: τον πηγαίο κώδικα με ή χωρίς το σώμα των υποπρογραμμάτων, τις επικεφαλίδες των υποπρογραμμάτων, τον κώδικα μιας μόνο διαδικασίας ή συνάρτησης, τη στοίβα των ενεργών υποπρογραμμάτων και τέλος τον «σκελετό» ενός προγράμματος με τη μορφή ενός δένδρου, στη ρίζα του οποίου βρίσκεται το κυρίως πρόγραμμα. Το Karel Genie

χρησιμοποιείται για περισσότερα από 10 χρόνια σε σχολεία και Πανεπιστήμια της Αμερικής. Μεταξύ άλλων έχει χρησιμοποιηθεί στο Harvard University, στο New York University και στο Stanford University. Πρόσφατα στο Πανεπιστήμιο Μακεδονίας άρχισε να αναπτύσσεται και μια ελληνική έκδοση του Karel ([27]).

Οι μικρόκοσμοι και οι μικρογλώσσες βρίσκουν ευρεία χρήση όχι μόνο σε αρχάριους προγραμματιστές αλλά γενικότερα σε περιπτώσεις στις οποίες η διδασκαλία επικεντρώνεται στις έννοιες και όχι στις συντακτικές ιδιαιτερότητες μιας γλώσσας προγραμματισμού ή την ευκολία δόμησης μιας πραγματικής εφαρμογής. Έτσι, για παράδειγμα, στο Πανεπιστήμιο της Grenoble, η Logo και άλλες συναφείς γλώσσες χρησιμοποιούνται ακόμη και σε προχωρημένα στάδια σπουδών ($2^{ο}$ και $3^{ο}$ έτος). Ακόμη, υπακούοντας στην ίδια προβληματική, αναφέρουμε το πρόσφατο παράδειγμα για διδασκαλία του τυπικού προγραμματισμού, με τη βοήθεια μια μικρογλώσσας – της μηχανής του Post [26].

Βελτίωση των Διαγνωστικών Δυνατοτήτων των Μεταγλωττιστών

Περιγραφή. Σύμφωνα με αρκετούς ερευνητές (μεταξύ των οποίων είναι και οι Freunds, Roberts [10], και Schorsch [20]), οι δυσκολίες και η απογοήτευση των σπουδαστών κατά την εισαγωγή τους στον προγραμματισμό είναι περισσότερο συνάρτηση του προγραμματιστικού περιβάλλοντος που χρησιμοποιείται και όχι της γλώσσας προγραμματισμού. Ένα από τα προβλήματα των προγραμματιστικών περιβαλλόντων, το σημαντικότερο σύμφωνα με τους παραπάνω ερευνητές, είναι το γεγονός ότι οι εμπορικοί μεταγλωττιστές στους οποίους βασίζονται τα περιβάλλοντα αυτά δεν παρέχουν την απαραίτητη στήριξη στους αρχάριους προγραμματιστές. Συνεπώς, για τη στήριξη των σπουδαστών κατά την εισαγωγή τους στον προγραμματισμό μπορεί να χρησιμοποιηθεί μια γλώσσα προγραμματισμού γενικού σκοπού και ένα σύνολο προβλημάτων επεξεργασίας αριθμών και συμβόλων, όπως υπαγορεύει η κλασική προσέγγιση διδασκαλίας του προγραμματισμού, αλλά το περιβάλλον προγραμματισμού πρέπει οπωσδήποτε να διαθέτει ένα μεταγλωττιστή με βελτιωμένες διαγνωστικές δυνατότητες.

Πλεονεκτήματα. Η υιοθέτηση της συγκεκριμένης προσέγγισης διδασκαλίας του προγραμματισμού στην ουσία δίνει λύση στα προβλήματα που παρουσιάζουν οι εμπορικοί μεταγλωττιστές –και αναφέρονται παρακάτω-, όταν αυτοί χρησιμοποιούνται στα πλαίσια ενός μαθήματος εισαγωγής στον προγραμματισμό:

- Τα μηνύματα λάθους συχνά δεν παρέχουν επαρκείς πληροφορίες, ενώ μερικές φορές είναι και παραπλανητικά (π.χ. εμφάνιση μηνυμάτων συντακτικών λαθών αρκετές γραμμές κάτω από τη γραμμή όπου βρίσκεται το λάθος)
- Τα μηνύματα λάθους συχνά δεν είναι κατανοητά, αφού χρησιμοποιούν όρους που δεν μπορεί να κατανοήσει ο αρχάριος προγραμματιστής.
- Δεν κάνουν καθόλου ή σχεδόν καθόλου έλεγχο για τον εντοπισμό λογικών λαθών.
- Οι αλληλεπιδραστικοί αποσφαλματωτές απαιτούν την κατανόηση από τους σπουδαστές «προχωρημένων» εννοιών, χωρίς αυτοί να είναι σε θέση να τις αφομοιώσουν.
- Παρέχουν ένα σύνολο επιλογών και ειδικών χαρακτηριστικών που όχι μόνο είναι άχρηστα, αλλά αυξάνουν και την πολυπλοκότητα του συστήματος περιορίζοντας την κατανόηση και την αυτοπεποίθηση του αρχάριου προγραμματιστή.

Εκπαιδευτικά εργαλεία. Ένα από τα πιο αντιπροσωπευτικά εργαλεία που αναπτύχθηκε στα πλαίσια της συγκεκριμένης προσέγγισης είναι το ***THETIS*** [10] που αναπτύχθηκε στο Τμήμα Επιστήμης Υπολογιστών του Stanford University όπου και χρησιμοποιείται στα μαθήματα CS1 και CS2. Το THETIS αποτελείται από ένα ερμηνευτή της ANSI C και ένα *γραφικό ενδιάμεσο* που παρέχει εύχρηστα εργαλεία ανάπτυξης, αποσφαλμάτωσης και οπτικοποίησης προγραμμάτων. Τα βασικότερα χαρακτηριστικά του είναι τα εξής:

- *Βελτίωση των μηνυμάτων λάθους*. Τα μηνύματα λάθους που αναφέρει το THETIS είναι ευνόητα και παρέχουν όλες τις απαραίτητες πληροφορίες. Για παράδειγμα, οι εντολές *char a; a = "c";* έχουν ως αποτέλεσμα την εμφάνιση του ακριβούς μηνύματος "cannot assign a value of type string to a variable of type char.", ενώ ένας εμπορικός μεταγλωττιστής θα εμφάνιζε

ένα μήνυμα του τύπου "type mismatch", το οποίο δεν παρέχει αρκετές πληροφορίες ή "assignment makes integer from pointer without cast", το οποίο εκφράζει το λάθος με τρόπο που ένας αρχάριος προγραμματιστής δεν μπορεί να κατανοήσει.

- *Ενδυνάμωση των συντακτικών περιορισμών*. Οι C και C++ λαμβάνουν ως έγκυρες συγκεκριμένες συντακτικές δομές οι οποίες είναι σχεδόν πάντα λανθασμένες όταν γράφονται από αρχάριους προγραμματιστές. Το πιο συνηθισμένο παράδειγμα είναι η χρήση του τελεστή απόδοσης τιμής "=" στη θέση του τελεστή συσχέτισης "= =". Για παράδειγμα, η εντολή " if (j = 0)…" η οποία είναι συντακτικά σωστή, έχει ως αποτέλεσμα την ανάθεση της τιμής 0 στη μεταβλητή j, που μεταφράζεται τελικά ως FALSE. Φυσικά, η πρόθεση του προγραμματιστή είναι ο έλεγχος ικανοποίησης της συνθήκης "j ίσο με 0" και για το λόγο αυτό το THETIS θεωρεί την εντολή λανθασμένη και εμφανίζει σχετικό μήνυμα. Οι περισσότεροι συντακτικοί περιορισμοί που έχουν ενσωματωθεί στο THETIS είναι παρόμοιοι με εκείνους που έχουν προτείνει οι δημιουργοί της ***Educational C*** (EC) [17], ενός υποσυνόλου της C για εκπαιδευτικούς σκοπούς.
- *Εντοπισμός λαθών εκτέλεσης*. Η πιο χρονοβόρα και δύσκολη φάση της αποσφαλμάτωσης είναι η εύρεση των λογικών λαθών ενός προγράμματος. Γι' αυτό το λόγο το THETIS ελέγχει για την ύπαρξη των εξής λαθών εκτέλεσης: διαίρεση με το 0, χρήση μη αρχικοποιημένης μεταβλητής, αναφορά ή αποδέσμευση μη έγκυρου δείκτη, εκτός ορίων προσπέλαση πίνακα, έξοδος, χωρίς επιστροφή τιμής, από μια συνάρτηση που επιστρέφει τιμή, πέρασμα μη έγκυρων ορισμάτων σε μια συνάρτηση, ανάθεση μη έγκυρης τιμής σε ένα βαθμωτό τύπο.
- Επιπλέον, το THETIS ενσωματώνει και κάποια *εργαλεία αποσφαλμάτωσης και οπτικοποίησης*.

Εικονικές Γλώσσες Προγραμματισμού

Περιγραφή. Οι υποστηρικτές (Calloni, Bagert [6], Studer, Taylor, Macie [23]) των εικονικών γλωσσών προγραμματισμού θεωρούν ότι η διδασκαλία του προγραμματισμού σε αρχάριους πρέπει να βασίζεται σε ένα εικονικό- βασισμένο σε διαγράμματα ροής περιβάλλον και όχι σε ένα περιβάλλον στο οποίο η εκφορά των αλγορίθμων είναι κειμενική. Αρκετοί από τους ερευνητές αυτούς χρησιμοποιούν ως απόδειξη των ισχυρισμών τους τα αποτελέσματα των πειραμάτων που διεξήγαγε ο ερευνητικός ψυχολόγος David A. Scanlan και τα οποία είχαν ως αντικείμενο την αποτελεσματικότητα των διαφόρων μεθόδων κατανόησης των αλγορίθμων. Σε ένα από τα εμπεριστατωμένα πειράματά του [19] ο Scanlan επισημαίνει ότι ενώ τόσο η γραφική αναπαράσταση των αλγορίθμων όσο και ο ψευδοκώδικας συντελούν ουσιαστικά στην κατανόηση των αλγορίθμων, ωστόσο, η γραφική αναπαράστασή τους είναι πιο αποτελεσματική για τις «διανοητικές διαδικασίες» που σχετίζονται με τη διδασκαλία και κατανόηση της διαδικασίας ανάπτυξης αλγορίθμων.

Πλεονεκτήματα. Ένα από τα βασικότερα πλεονεκτήματα των εικονικών γλωσσών προγραμματισμού είναι η ελαχιστοποίηση των συντακτικών λεπτομερειών με τις οποίες έρχεται αντιμέτωπος ο σπουδαστής. Η διδασκαλία του προγραμματισμού στα πλαίσια της προσέγγισης αυτής επικεντρώνεται στην ανάπτυξη ικανοτήτων επίλυσης προβλημάτων - ανάπτυξης αλγορίθμων και όχι στην εκμάθηση μιας γλώσσας προγραμματισμού, χωρίς αυτό να είναι κανόνας. Τέλος, όπως ήδη αναφέρθηκε χρησιμοποιώντας μια εικονική γλώσσα μειώνεται δραματικά η διανοητική πολυπλοκότητα που απαιτεί η εκφορά ενός αλγορίθμου σε μια κειμενική γλώσσα προγραμματισμού.

Εκπαιδευτικά εργαλεία. Η πρώτη αξιόλογη εικονική γλώσσα προγραμματισμού αναπτύχθηκε στο Texas Tech University το 1992 και ονομάζεται ***BACCII*** [6]. Στην ουσία πρόκειται για ένα περιβάλλον προγραμματισμού που επιτρέπει στο χρήστη να προγραμματίζει χρησιμοποιώντας εικόνες που αναπαριστάνουν όλες τις βασικές δομές & τύπους δεδομένων. Για την ανάπτυξη των αλγορίθμων χρησιμοποιείται το *συντακτικά καθοδηγούμενο ενδιάμεσο* (syntax directed interface) του BACCII που επιτρέπει την εισαγωγή στον αναπτυσσόμενο αλγόριθμο, που έχει τη μορφή ενός διαγράμματος ροής, μόνο συντακτικά σωστών εντολών (εικόνων). Οι απαραίτητες αλλαγές στο διάγραμμα (π.χ. συνδέσεις) πραγματοποιούνται

αυτόματα. Μετά το τέλος της ανάπτυξης ενός αλγορίθμου υπάρχει η δυνατότητα αυτόματης παραγωγής συντακτικά σωστού πηγαίου κώδικα σε Pascal, C, Fortran, & Basic. Την BACCII διαδέχτηκε η ***BACCII++*** [7] που έχει ως στόχο τη στήριξη της διδασκαλίας εννοιών και γλωσσών τόσο του διαδικαστικού όσο και του αντικειμενοστραφούς παραδείγματος προγραμματισμού. Συγκεκριμένα, η BACCII++ έχει ως στόχο τη στήριξη των μαθημάτων "εισαγωγή στον προγραμματισμό" (CS1462) και "δομές δεδομένων/αντικειμενοστραφής προγραμματισμός" (CS2463), χρησιμοποιώντας τη C++. Η μοναδική ουσιαστική διαφορά του περιβάλλοντος BACCII++ σε σχέση με το BACCII είναι η ανάπτυξη ενός *αλληλεπιδραστικού πολυμεσικού συστήματος* για τη στήριξη των εργαστηρίων.

Συστήματα Δυναμικής Προσομοίωσης Εκτέλεσης Προγραμμάτων

Περιγραφή. Η προσέγγιση αυτή, αλλά και οι προσεγγίσεις που ακολουθούν, βασίζονται στην τεχνολογία της οπτικοποίησης λογισμικού που εμφανίστηκε το 1981 στο 30-λεπτο φιλμ Sorting Out Sorting [1] του Ron Baecker. Το φιλμ αυτό, το οποίο έχει χρησιμοποιηθεί σε αρκετά Πανεπιστήμια και σχολεία για τη διδασκαλία των εννέα αλγορίθμων ταξινόμησης που περιλαμβάνει, αποτελεί σταθμό στην ιστορία της οπτικοποίησης λογισμικού. Ο όρος *οπτικοποίηση λογισμικού* (software visualization) αναφέρεται στη χρήση τεχνικών της τυπογραφίας, του σχεδιασμού γραφικών, της δυναμικής προσομοίωσης (animation) και της κινηματογραφίας, καθώς και των σύγχρονων τεχνολογιών της αλληλεπίδρασης ανθρώπου-υπολογιστή και των γραφικών υπολογιστή με σκοπό τη στήριξη στην κατανόηση εννοιών και την αποτελεσματική χρήση του λογισμικού των υπολογιστών [16]. Η *οπτικοποίηση προγράμματος* (program visualization) ειδικότερα αναφέρεται στην οπτικοποίηση του κώδικα ή/και των δομών δεδομένων ενός προγράμματος και μπορεί να είναι είτε *στατική* (π.χ. παρουσίαση μιας συνδετικής λίστας με ένα διάγραμμα), είτε δυναμική [16]. Το πιο απλό παράδειγμα δυναμικής οπτικοποίησης κώδικα είναι το μαρκάρισμα της γραμμής του κώδικα που εκτελείται. Τα συστήματα που έχουν δυνατότητες δυναμικής προσομοίωσης κώδικα αναφέρονται συχνά ως *δυναμικοί προσομοιωτές εκτέλεσης προγραμμάτων* (program animators).

Πλεονεκτήματα. Τα πλεονεκτήματα της συγκεκριμένης προσέγγισης, αλλά και της οπτικοποίησης λογισμικού γενικότερα, είναι πολλά [2]:

- Κατανόηση πολύπλοκων εννοιών μέσω της χρήσης εικόνων, εναλλακτικός τρόπος παρουσίασης εννοιών, διατήρηση της προσοχής των διδασκόμενων, διδασκαλία περισσότερης ύλης σε λιγότερο χρόνο και με λιγότερα λάθη, αύξηση του βαθμού κατανόησης.
- Παρέχει τη δυνατότητα κατανόησης θεωρητικών εννοιών μέσω πρακτικής εξάσκησης, όπως για παράδειγμα πειραματισμού με διαφορετικά δεδομένα και εξερεύνησης ποικίλων απόψεων ενός προβλήματος, μιας έννοιας ή ενός αλγορίθμου με το ρυθμό που επιθυμεί ο χρήστης.
- Απλοποιεί την διαδικασία της αποσφαλμάτωσης των προγραμμάτων.
- Επιτρέπει την εύκολη παρουσίαση πολύπλοκων δομών (π.χ. γραφημάτων) με ακρίβεια.

Εκπαιδευτικά Εργαλεία. Το πιο κλασικό παράδειγμα δυναμικού προσομοιωτή εκτέλεσης προγραμμάτων είναι το ***DYNALAB***. Το DYNALAB Version 3.0 [3] είναι η τελευταία έκδοση μιας σειράς δυναμικών προσομοιωτών εκτέλεσης προγραμμάτων, η ανάπτυξη των οποίων ξεκίνησε στις αρχές της δεκαετίας του '80 στο Τμήμα Επιστήμης Υπολογιστών του Montana State University και εξακολουθεί μέχρι σήμερα να αποτελεί ένα ενεργό project. Το Dynalab αποτελείται από έναν ερμηνευτή της Turbo Pascal και ένα γραφικό ενδιάμεσο που παρέχει στο χρήστη τη δυνατότητα εκτέλεσης ενός προγράμματος βήμα προς βήμα, είτε προς τα εμπρός είτε προς τα πίσω με το ρυθμό που επιθυμεί ο χρήστης. Κάθε εντολή πριν να εκτελεστεί μαρκάρεται, ενώ η εκτέλεσή της έχει ως αποτέλεσμα την εμφάνιση/ενημέρωση των τιμών των μεταβλητών-σταθερών-παραμέτρων σε ένα τμήμα της οθόνης και την εμφάνιση της εισόδου/εξόδου του προγράμματος σε ένα άλλο τμήμα της. Επίσης, ενημερώνεται και ο μετρητής των εκτελούμενων εντολών που είναι ιδιαίτερα χρήσιμος για

την ανάλυση της πολυπλοκότητας των αλγορίθμων που αναπτύσσει ο χρήστης ή/και τη σύγκριση της πολυπλοκότητας διαφορετικών αλγορίθμων για το ίδιο πρόβλημα.

Χρήση Τεχνικών Δυναμικής Προσομοίωσης Εκτέλεσης Αλγορίθμων

Περιγραφή. Η προσέγγιση αυτή, που έχει προταθεί από τους Mukherjia και Stasko [12], [13], [22], προτείνει τη χρήση τεχνικών *δυναμικής προσομοίωσης εκτέλεσης αλγορίθμων* για την ανίχνευση, αποσφαλμάτωση και κατανόηση των προγραμμάτων. Συγκεκριμένα, οι Mukherjia και Stasko προτείνουν το συνδυασμό των εκπαιδευτικών τεχνολογιών της δυναμικής προσομοίωσης εκτέλεσης προγραμμάτων και της δυναμικής προσομοίωσης εκτέλεσης αλγορίθμων για τη δημιουργία ενός συστήματος που θα εκμεταλλεύεται τα πλεονεκτήματα και των δύο. Η προβληματική των Mukherjia και Stasko, η οποία καθοδηγεί και την ανάπτυξη του συστήματος LENS στο οποίο θα αναφερθούμε στη συνέχεια, επικεντρώνεται σε δύο σημεία:

- Οι προγραμματιστές πολύ συχνά κατά την ανάπτυξη προγραμμάτων δημιουργούν νοητά μοντέλα του προγράμματός τους, του τρόπου λειτουργίας του και χρήσης των δεδομένων. Αυτή η γενική εικόνα ενός προγράμματος είναι ιδιαίτερα χρήσιμη για την κατανόηση του τρόπου συμπεριφοράς των διάφορων διαδικασιών και των δεδομένων σε συνδυασμό μεταξύ τους. Τα συστήματα δυναμικής προσομοίωσης εκτέλεσης προγραμμάτων, αν και παρέχουν αυτόματα πολλές πληροφορίες κατά την εκτέλεση ενός προγράμματος δεν παρέχουν μια συνολική εικόνα.
- Τη συνολική εικόνα ενός προγράμματος, ή καλύτερα την οπτική παρουσίαση του νοητού μοντέλου του, μπορεί να την προσφέρει η δυναμική προσομοίωση εκτέλεσης αλγορίθμων. Το πρόβλημα είναι ότι η δημιουργία δυναμικών προσομοιώσεων της εκτέλεσης αλγορίθμων είναι μια διαδικασία χρονοβόρα, η οποία απαιτεί την εκμάθηση και χρήση ενός πακέτου γραφικών και η οποία συνήθως περιορίζεται σε προγράμματα που είναι ήδη σωστά. Μια τέτοια διαδικασία όμως δεν μπορεί να χρησιμοποιηθεί στην ανάπτυξη και αποσφαλμάτωση προγραμμάτων, στις οποίες ο χρόνος είναι σημαντικός παράγοντας.

Πλεονεκτήματα. Τα βασικότερα πλεονεκτήματα συνοψίζονται ως εξής:

- Παρέχει μια συνολική εικόνα ενός προγράμματος.
- Παρέχει τη δυνατότητα δημιουργίας εξεικονίσεων, αναπαράστασης δηλαδή των μεταβλητών με εικόνες, με εύκολο τρόπο.
- Οι εξεικονισμένες μεταβλητές έχουν δυναμική υπόσταση, μιας και κατά την εκτέλεση ενός προγράμματος οι εικόνες που τους αντιστοιχούν μεταβάλλονται (χρώμα, μέγεθος κ.τ.λ.) σε σχέση με τη μεταβολή των τιμών των μεταβλητών.

Εκπαιδευτικά Εργαλεία. Το μοναδικό εκπαιδευτικό εργαλείο που έχει αναπτυχθεί στα πλαίσια της συγκεκριμένης προσέγγισης είναι το ***LENS*** ([12], [13], [22]). Το LENS υλοποιήθηκε σε περιβάλλον UNIX και χρησιμοποιείται για τον έλεγχο και την αποσφαλμάτωση προγραμμάτων σε C. Για την παρουσίαση των δυναμικών προσομοιώσεων χρησιμοποιείται το σύστημα δυναμικής προσομοίωσης εκτέλεσης αλγορίθμων XTango. Η δημιουργία των δυναμικών προσομοιώσεων πραγματοποιείται μέσω τεχνικών *άμεσης διαχείρισης* (direct manipulation) και *προγραμματισμού που πραγματοποιείται παραδειγματικά* (demonstration). Οι διαδικασίες δυναμικής προσομοίωσης έχουν περιοριστεί σε μεγάλο βαθμό: μετακίνηση αντικειμένου, αλλαγή χρώματος αντικειμένου, αλλαγή του στυλ γεμίσματος, φωτισμός ενός αντικειμένου, ανταλλαγή της θέσης δύο αντικειμένων και διαγραφή ή απόκρυψη ενός αντικειμένου. Η επιλογή των αντικειμένων και διαδικασιών έγινε μετά από τη μελέτη 42 δυναμικών προσομοιώσεων εκτέλεσης αλγορίθμων, οι οποίες δημιουργήθηκαν από 25 διαφορετικά άτομα χρησιμοποιώντας το σύστημα XTango.

Δυναμική Ηχητική Προσομοίωση της Εκτέλεσης Προγραμμάτων

Περιγραφή. Ο όρος *δυναμική ηχητική προσομοίωση εκτέλεσης προγραμμάτων* (program auralization) αναφέρεται στην οπτικοποίηση του λογισμικού μέσω του ήχου. Συγκεκριμένα, είναι η διαδικασία δημιουργίας νοητών εικόνων της συμπεριφοράς, δομής και λειτουργίας των προγραμμάτων ακούγοντας ηχητικές αναπαραστάσεις της εκτελέσεώς τους. Σε αντίθεση

δηλαδή με τα κλασικά συστήματα οπτικοποίησης λογισμικού, τα οποία χρησιμοποιούν κείμενο ή γραφικά, στη συγκεκριμένη περίπτωση χρησιμοποιούνται μουσικοί τόνοι, λόγος και ηχογραφημένες συνθέσεις ήχων για την καλύτερη κατανόηση των προγραμμάτων [4].

Πλεονεκτήματα. Οι ερευνητές DiGiano, Baecker, Marcus [8], Francioni, Albright και Jackson [9] αναφέρουν αρκετούς λόγους για τη χρήση του ήχου ως μέσου οπτικοποίησης:

- Η οπτικοποίηση είναι σε μεγάλο βαθμό υποκειμενική, και ότι είναι χρήσιμο για κάποιο άτομο είναι επουσιώδες για κάποιο άλλο. Ο ήχος παρέχει ένα ακόμη μέσο οπτικοποίησης ενός προγράμματος, το οποίο μπορεί να αποδειχτεί ιδιαίτερα χρήσιμο για κάποια άτομα.
- Παρέχει βοήθεια κατά την αποσφαλμάτωση των προγραμμάτων, χωρίς να επεμβαίνει στο γραφικό ενδιάμεσο.
- Ο ήχος είναι ίσως πιο κατάλληλος από τη χρήση γραφικών μεθόδων για την παρουσίαση συγκεκριμένων τύπων πληροφορίας (π.χ. επαναληπτικές δομές).
- Η χρήση του ήχου παρέχει ένα ισχυρό μέσο για την παρακολούθηση της κατάστασης μεγάλων ποσοτήτων δεδομένων.
- Δίνει λύση στο πρόβλημα του προγραμματισμού σε συσκευές με περιορισμένο μέγεθος οθόνης.
- Η παρακολούθηση του ήχου μπορεί να γίνει παθητικά. Αυτό σημαίνει ότι δεν χρειάζεται κανείς να είναι ιδιαίτερα συγκεντρωμένος για να αντιληφθεί ότι κάτι ασυνήθιστο συμβαίνει κατά τη εκτέλεση ενός προγράμματος. Επιπλέον, οι προγραμματιστές έχουν τη δυνατότητα να αλληλεπιδρούν ταυτόχρονα με το γραφικό ενδιάμεσο.

Οι Francioni, Albright και Jackson ήταν από τους πρώτους ερευνητές που χρησιμοποίησαν τον ήχο ως μέσο οπτικοποίησης παράλληλων προγραμμάτων. Τα πειράματά τους επιβεβαίωσαν την υπόθεση ότι η χρήση του ήχου θα βοηθούσε στην παρακολούθηση της συμπεριφοράς και στην αποσφαλμάτωση παράλληλων προγραμμάτων. Επισημάνθηκε ωστόσο ότι η χρήση του ήχου δεν θα έπρεπε να αντικαταστήσει τις κλασικές πλέον μεθόδους οπτικοποίησης, αλλά να τις εμπλουτίσει ακόμα περισσότερο [4].

Εκπαιδευτικά Εργαλεία. Ένα από τα πρώτα project που αναπτύχθηκαν με σκοπό να διαπιστωθεί ο βαθμός στον οποίο η μουσική ανάδραση μπορεί να βοηθήσει τους αρχάριους προγραμματιστές στην αποσφαλμάτωση των προγραμμάτων τους, ήταν το project ***CAITLIN*** [24]. Το CAITLIN είναι ένας προεπεξεργαστής της Pascal, ο οποίος επιτρέπει στους χρήστες να αντιστοιχίσουν ήχους σε κάθε δομή ελέγχου ενός προγράμματος. Ωστόσο, οι χρήστες δεν έχουν τη δυνατότητα αντιστοίχισης ήχου σε γραμμές κώδικα, κλήσεις διαδικασιών ή τη ροή των δεδομένων. Η αντιστοίχιση ήχου σε κάθε δομή αποτελείται από τρία μέρη: ένα ήχο που σηματοδοτεί την είσοδο σε μια δομή, μια σύνθεση που αναπαριστά την εκτέλεση της και ένα ήχο που σηματοδοτεί την έξοδο από αυτή. Προκαταρκτικά πειράματα απέδειξαν ότι οι χρήστες μπορούσαν να παρακολουθήσουν την εκτέλεση απλών προγραμμάτων, να αναγνωρίσουν δηλαδή τις βασικές δομές τους. Ωστόσο, οι χρήστες ξεχνούσαν συχνά τον ήχο που είχαν αντιστοιχίσει σε κάθε δομή [4].

Επίλογος

Όπως έγινε φανερό από τις προηγούμενες ενότητες, έχουν γίνει πολλές προσπάθειες για την ανάπτυξη ειδικών μεθοδολογιών, γλωσσών (και μικρών-γλωσσών) και εργαλείων για τη στήριξη της διδασκαλίας του προγραμματισμού. Στην εργασία αυτή, που αποτελεί προϊόν συγκριτικής μελέτης και σχολιασμού του μεγάλου όγκου στοιχείων που είναι διάσπαρτα στη βιβλιογραφία για τη διδασκαλία/εκμάθηση του προγραμματισμού και τα εκπαιδευτικά εργαλεία για τη στήριξή της, ταξινομήσαμε τις προσπάθειες και συνοψίσαμε τις εμπειρίες αυτές. Από την επισκόπηση των προσεγγίσεων διδασκαλίας του προγραμματισμού και των εκπαιδευτικών εργαλείων που έχουν αντίστοιχα αναπτυχθεί, γίνεται προφανές, ότι όλες οι εναλλακτικές προσεγγίσεις διδασκαλίας του προγραμματισμού δίνουν λύση σε ένα ή περισσότερα από τα διδακτικά προβλήματα που παρουσιάζονται όταν η εισαγωγή στον προγραμματισμό πραγματοποιείται με την κλασική προσέγγιση. Ωστόσο, υπάρχει ανάγκη

αξιολόγησης των διάφορων προσεγγίσεων διδασκαλίας του προγραμματισμού και των εκπαιδευτικών τεχνολογιών – εργαλείων που έχουν αναπτυχθεί με σκοπό τη στήριξης της διδακτικής πράξης, προκειμένου η διδασκαλία – εκμάθηση του προγραμματισμού να γίνει πιο ουσιαστική και αποτελεσματική.

Ιδιαίτερο ενδιαφέρον τόσο από ερευνητική όσο και από διδακτική άποψη πιστεύουμε ότι παρουσιάζει η μελέτη της εφαρμογής διαφορετικών προσεγγίσεων της διδασκαλίας του προγραμματισμού στη χώρας μας. Από την έρευνα που έχουμε διενεργήσει στη διεθνή βιβλιογραφία, πιστεύομε ότι η μέχρι τώρα επιτυχημένη χρήση των μικρόκοσμων προγραμματισμού και ιδιαίτερα του Robot Karel σε διάφορα εκπαιδευτικά ιδρύματα του εξωτερικού αποτελεί μια καλή πρόταση για χρήση του μικρόκοσμου αυτού για τη διδασκαλία του προγραμματισμού και στα σχολεία της Β'θμιας εκπαίδευσης της χώρας μας. Η διδασκαλία του προγραμματισμού στη Β'θμια εκπαίδευση θα μπορούσε να ξεκινάει βασιζόμενη στον μικρόκοσμο αυτόν. Μ' αυτόν τον τρόπο η διδασκαλία του προγραμματισμού θα επικεντρωθεί σε έννοιες και όχι σε συντακτικές ιδιαιτερότητες μιας γλώσσας προγραμματισμού, καθώς και στην ανάπτυξη δεξιοτήτων για την αλγοριθμική επίλυση των προβλημάτων. Στη συνέχεια θα μπορούσε να γίνει χρήση ενός Συστήματος Δυναμικής Προσομοίωσης Εκτέλεσης Προγράμματος, όπως για π.χ. του Dynalab. Η χρήση του Dynalab στα Ελληνικά σχολεία της Β'θμιας εκπαίδευσης μπορεί να δώσει λύση στα προβλήματα που οφείλονται στη χρήση των επαγγελματικών περιβαλλόντων προγραμματισμού που όπως έχει ήδη διαπιστωθεί είναι ένας σημαντικός παράγοντας δυσκολίας για τους αρχάριους. Ακόμη η δυνατότητα της απόκτησης του Dynalab δωρεάν από το διαδίκτυο, η χρήση ενός ερμηνευτή της Turbo Pascal (γλώσσα που διδάσκεται στο Λύκειο) από το Dynalab και η διάθεση από τους δημιουργούς ενός μεγάλου αριθμού λυμένων ασκήσεων – προγραμμάτων στο διαδίκτυο, θεωρούμε ότι αποτελεί μια καλή λύση για τα σχολεία της Β'θμιας ή ακόμη και τριτοβάθμιας εκπαίδευσης της χώρας μας.

Βιβλιογραφία

[1] Baecker, R. (1997) Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Science. In *Software Visualization: Programming as a multimedia Experience,* Stasko, J., Domingue, J., Brown, M. and Price, B., Eds. MIT Press, Cambridge, σελ. 369-381.

[2] Bergin, J., Brodlie, K., Goldweber, M., Jimenez-Peris, R., Khuri, S., Patino-Martinez, M., McNally, M., Naps, T., Rodger, S. & Wilson, J. (1996) An overview of visualization: its use and design - Report of the Working Group on Visualization. *ACM, Integrating Tech. Into C.S.E 6/96 Barselona, Spain*, σελ. 192-200.

[3] Birch, M., Boroni, C., Goosey, F., Patton, S., Poole, D., Pratt, C. & Ross, R. (1995) DYNALAB: A Dynamic Computer Science Laboratory Infrastructure Featuring Program Animation. *ACM, SIGSCE Bulletin*, σελ. 29-33 & *http://www.cs.montana.edu/~dynalab.*

[4] Brown, M. και Hershberger, J. (1997) Program Auralization. In Software Visualization: *Programming as a multimedia Experience,* Stasko, J., Domingue, J., Brown, M. and Price, B., Eds. MIT Press, Cambridge, σελ. 139-144.

[5] Brusilovski, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller P. (1997) Mini-languages: a way to learn programming principles. *Education and Information Technologies 2*, σελ. 65-83.

[6] Calloni, B. & Bagert, D. (1994) Iconic Programming in BACCII vs. Textual Programming: which is a better learning environment? *ACM, SIGSCE '94 3/94, Phoenix AZ*, σελ. 188-192.

[7] Calloni, B. A. & Bagert, D. J. (1997) Iconic Programming Proves Effective for Teaching the First Year Programmming Sequence., *ACM*, *SIGSCE '97 CA, USA,* σελ. 262-266.

[8] DiGiano, C., Baecker, R. & Marcus, A. (1997) Software visualization for Debugging. *Communications of the ACM, Vol. 40, No. 4*, σελ. 44-54.44

[9] Francioni, J., Albright, L. & Jackson, J. (1991) Debugging parallel programs using sound. *ACM, SIGPLAN Notices, Vol.26, No.12*, σελ. 68-75.

[10] Freund, S. N. & Roberts, E. S. (1996) THETIS: An ANSI C programming environment designed for introductory use., ACM, SIGSCE '96 2/96 Philadelphia, PA USA, σελ. 300-304.

[11] MacGNOME Project, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213.

[12] Mukherjia, S. & Stasko, J. (1993) Applying Algorithm Animation Techniques for Program Tracing, Debugging, and Understanding. 1993 IEEE 0279-5257/93, σελ. 456-465.

[13] Mukherjia, S. & Stasko, J. (1994) Toward Visual Debugging: Integrating Algorithm Animation Capabilities within a Source-Level Debugger. *ACM Transacions on Computer-Human Interaction, Vol. 1, No. 3, September 1994*, σελ. 215-244.

[14] Pane, J.F. & Myers, B.A. (1996) Usability Issues in the Design of Novice Programming Systems, Technical Report CMU-CS-96-132, School of Computer Science, Carnegie Mellon University.

[15] Pattis, R. E., Roberts, J. & Stehlik, M. (1995) Karel - The Robot, A Gentle Introduction to the Art of Programming. 2nd edn. New York, Wiley.

[16] Price, B., Baecker, R. & Small, I. (1997) An Introduction to Software Visualization. In *Software Visualization: Programming as a multimedia Experience,* Stasko, J., Domingue, J., Brown, M. and Price, B., Eds. MIT Press, Cambridge, σελ. 3-28.

[17] Ruckert, M. & Halpern, R. (1993) Educational C. *ACM*, *SIGSCE Bulletin*, σελ. 6-9.

[18] Sangwan, R. S., Korsh, J. F. & LaFollette, P. S. (1998) A System for Program Visualization in the Classroom., *ACM, SIGSCE '98 Atlanta GA USA,* σελ. 272-276.

[19] Scanlan, D. (1989) Structured flowharts outperform pseudocode:an experimental comparison, *IEEE Software, Vol 6, No 5, Sept. 1989*, σελ. 28-36.

[20] Schorsch, T. (1995) CAP: An automated self-assessment tool to check Pascal programs for syntax, logic and style errors. *ACM, SIGSCE '95 3/95 Nashville,USA*, σελ. 168-172.

[21] Smith, D.C., Cypher, A. & Sprohrer, J. (1994) KIDSIM: Programming Agents Without a Programming Language. *Communications of the ACM, Vol.37, No.7*, σελ. 55-67.

[22] Stasko, J. (1997) Building software Visualizations through Direct Manipulation and Demonstration. *In Software Visualization: Programming as a multimedia Experience*, Stasko, J., Domingue, J., Brown, M. and Price, B., Eds. MIT Press, Cambridge, σελ. 187-203.

[23] Studer, S., Taylor, J. & Macie, K. (1995) YOUNGSTER: A simplified introduction to computing removing the details so that a child may program. *ACM, SIGSCE '95* 3/95 *Nashville, TN USA*, σελ. 102-105.

[24] Vickers, P. & Alty, J. (1996) CAITLIN: A Musical Program Auralization Tool to Assist Novice Programmers with Debugging. *Proceedings of ICAD 96*, http://www.santafe.edu/~kramer/icad/ICAD96/proc96/ vickers.htm.

[25] Δαγδιλέλης, Β. (1996) Διδακτική της πληροφορικής. Η διδασκαλία του προγραμματισμού: αντιλήψεις των σπουδαστών για την κατασκευή κι επικύρωση προγραμμάτων και διδακτικές καταστάσεις για τη διαμόρφωσή τους *Διδακτορική διατριβή*, Τμήμα Εφ. Πληροφορικής Πανεπιστήμιο Μακεδονίας.

[26] Β. Δαγδιλέλης, Μ. Σατρατζέμη, Η μηχανή του Post: ένας διδακτικός μικρόκοσμος για την εισαγωγή στον τυπικό προγραμματισμό, *Πρακτικά 6ου Πανελλήνιου Συνεδρίου Πληροφορικής*, Αθήνα, Δεκέμβριος, 1997, 193-203.

[27] Σατρατζέμη Μ., Δαγδιλέλης Β., Μαργαρίτης Κ., Μιχαήλ Σ., Παπανικολάου Σ. (1999) Ένας διδακτικός μικρόκοσμος για μια εισαγωγή στον προγραμματισμό: το ρομπότ Karel, *4ο Πανελλήνιο Συνέδριο με Διεθνή Συμμετοχή, Διδακτική των Μαθηματικών και Πληροφορική στην Εκπαίδευση*, Ρέθυμνο.