AnimPascal: Ένα Εκπαιδευτικό Περιβάλλον για τη Στήριξη Εισαγωγικών Μαθημάτων Προγραμματισμού

Μ. Σατρατζέμη
Επίκουρος Καθηγήτρια, Τμ. Εφαρμ. Πληροφορικής, Παν. Μακεδονίας
maya@uom.gr
Κ. Χατζηαθανασίου
Υποψήφια Διδάκτωρ, Τμ. Εφαρμ. Πληροφορικής, Παν. Μακεδονίας
hatzidin@uom.gr
Β. Δαγδιλέλης
Λέκτορας, Τμ. Νηπιαγωγών, Παιδαγωγική Σχολή Φλώρινας, ΑΠΘ
dagdil@uom.gr

Περίληψη
Στην εργασία αυτή παρουσιάζουμε ένα ολοκληρωμένο περιβάλλον προγραμματισμού, το AnimPascal. Πρόκειται για ένα Σύστημα Δυναμικής Προσομοίωσης Εκτέλεσης Προγράμματος (program animator) το οποίο ενσωματώνει επιπλέον τη δυνατότητα καταγραφής των ενεργειών των σπουδαστών (recordability). Ο στόχος του AnimPascal είναι διπλός, αφενός να βοηθήσει τους αρχάριους στη φάση της ανάπτυξης, της αποσφαλμάτωσης, της εκτέλεσης και ελέγχου ενός προγράμματος αλλά και να καταγράψει τις προσπάθειες τους ώστε να βοηθήσει τον διδάσκοντα στον εντοπισμό των λανθασμένων αντιλήψεων τους. Στην εργασία αυτή παρουσιάζονται οι λειτουργίες του συστήματος, μερικά αποτελέσματα από τη χρήση του AnimPascal και οι αντιλήψεις των σπουδαστών και τέλος τα συμπεράσματα από τη μέχρι τώρα χρήση του AnimPascal στην υποστήριξη της διδασκαλίας του προγραμματισμού.
Λέξεις κλειδιά: Εκπαιδευτικό λογισμικό, Σύστημα Δυναμικής Προσομοίωσης Εκτέλεσης Προγραμμάτων (program animator), εγγραψιμότητα (recordability), αντιλήψεις των σπουδαστών για την κατασκευή προγραμμάτων

Abstract
This paper describes AnimPascal, an integrated programming environment. AnimPascal is a program animator that incorporates the possibility of recording students' actions. The aim of AnimPascal is to help students understand the phases of developing, debugging, executing and verifying a program; and also, by recording the different versions of students' programs, to help teachers find out students' errors. In this paper we present the functions of the system and some empirical results concerning students' conceptions when they try to solve a problem of an algorithmic or programming nature. Finally, we give the conclusions regarding the use of AnimPascal in teaching programming.

1. Εισαγωγή

Τις τελευταίες δεκαετίες εκτεταμένες έρευνες έχουν γίνει για τη μελέτη των δυσκολιών που αντιμετωπίζουν οι αρχάριοι σπουδαστές όταν διδάσκονται τις εισαγωγικές έννοιες του προγραμματισμού [3, 6, 10]. Αποτέλεσμα αυτών των ερευνών είναι η ανακάλυψη σημαντικών παραγόντων που καθιστούν την εκμάθηση του προγραμματισμού δύσκολη, που δικαιολογούν την αποτελεσματικότητα ή όχι των γλωσσών και των ολοκληρωμένων περιβαλλόντων προγραμματισμού που υπάρχουν, καθώς και των διαφόρων μεθόδων διδασκαλίας.

Μερικές από τις δυσκολίες που εντοπίστηκαν στους αρχάριους προγραμματιστές συνοψίζονται στις παρακάτω κατηγορίες:

- Δυσκολίες που προκύπτουν από την κατανόηση των γενικών ιδιοτήτων της «νοητής μηχανής» που μαθαίνει να ελέγχει ο αρχάριος και της σχέσης της με τη φυσική μηχανή.
- Δυσκολίες που οφείλονται στη σύνταξη και τη σημασιολογία των γλωσσών προγραμματισμού, οι οποίες θεωρείται ότι αποτελούν επέκταση των ιδιοτήτων και της συμπεριφοράς της νοητής μηχανής.

- Η εκμάθηση των καθιερωμένων δομών.
- Η απόκτηση της ικανότητας καθορισμού, ανάπτυξης, ελέγχου και αποσφαλμάτωσης ενός προγράμματος με τα διαθέσιμα εργαλεία.
- Οι συντάκτες προγραμμάτων (editors), μεταγλωττιστές (compilers) και αποσφαλματωτές (debuggers) έχουν δημιουργηθεί για επαγγελματίες προγραμματιστές και δημιουργούν επιπλέον εμπόδια στους αρχάριους προγραμματιστές.
- Τα περιβάλλοντα προγραμματισμού δεν διαθέτουν δυνατότητες οπτικοποίησης της εκτέλεσης ενός προγράμματος. Έτσι η διαδικασία της εκτέλεσης παραμένει «κρυμμένη» μ' αποτέλεσμα οι σπουδαστές να αναπτύσσουν μια αντίληψη για τη φάση της εκτέλεσης προσανατολισμένη περισσότερο σε Είσοδο / Έξοδο δεδομένων. Μ' αυτό τον τρόπο η έλλειψη οπτικής ανάδρασης δεν βοηθά στην κατανόηση της σημασιολογίας της γλώσσας.

Λαμβάνοντας υπόψη τα παραπάνω αναπτύξαμε ένα ολοκληρωμένο περιβάλλον προγραμματισμού, το AnimPascal[1], το οποίο και παρουσιάζουμε στην εργασία αυτή. Πρόκειται για ένα Σύστημα Δυναμικής Προσομοίωσης Εκτέλεσης Προγραμμάτων (program animator) [8], το οποίο ενσωματώνει επιπλέον τη δυνατότητα καταγραφής των ενεργειών των σπουδαστών (recordability). Ο στόχος του AnimPascal είναι διπλός, αφενός να βοηθήσει τους αρχάριους στη φάση της ανάπτυξης, αποσφαλμάτωσης και εκτέλεσης του προγράμματος αλλά και να βοηθήσει τον διδάσκοντα στον εντοπισμό των λανθασμένων αντιλήψεων των σπουδαστών.

Στις επόμενες ενότητες παρουσιάζονται οι λειτουργίες του συστήματος, μερικά αποτελέσματα από τη χρήση του AnimPascal και τις αντιλήψεις των σπουδαστών και τέλος τα συμπεράσματα από τη μέχρι τώρα χρήση του AnimPascal στην υποστήριξη της διδασκαλίας του προγραμματισμού.

2. Περιγραφή λειτουργιών του AnimPascal

Τα αποτελέσματα των ερευνών σχετικά με τις δυσκολίες της διδασκαλίας και εκμάθησης του προγραμματισμού και η πρόοδος της τεχνολογίας των υπολογιστικών συστημάτων, οδήγησαν στην ανάπτυξη περιβαλλόντων προγραμματισμού που υποστηρίζουν τη διδασκαλία του αποτελεσματικότερα. Η άποψή μας είναι ότι η ανάπτυξη συστημάτων προγραμματισμού για αρχάριους πρέπει να γίνεται κατά τέτοιο τρόπο ώστε το εκπαιδευτικό λογισμικό να αποτελεί ένα αποτελεσματικό εργαλείο για την επίτευξη ενός συγκεκριμένου διδακτικού στόχου [5]. Αν και κάποιοι θεωρούν αυτονόητη την ύπαρξη ενός ή περισσότερων διδακτικών στόχων κατά την ανάπτυξη εκπαιδευτικού λογισμικού, δεν είναι λίγες οι περιπτώσεις στις οποίες οι διδακτικοί στόχοι απουσιάζουν ή δεν είναι σαφείς. Αρκετά συστήματα προγραμματισμού για αρχάριους παρουσιάζουν ως στόχο τους την αντιμετώπιση των προβλημάτων των αρχάριων προγραμματιστών. Τα προβλήματα όμως των αρχάριων προγραμματιστών είναι ποικίλα, όπως και οι τρόποι αντιμετώπισής τους, σύμφωνα με σχετικές μελέτες, οι οποίες τις περισσότερες φορές δεν λαμβάνονται υπόψη. Επίσης, υπάρχει η λανθασμένη αντίληψη ότι η χρήση της τεχνολογίας από μόνη της έχει ως αποτέλεσμα τη δημιουργία αποτελεσματικών εκπαιδευτικών εργαλείων.

Η εκπαιδευτική εφαρμογή AnimPascal σκοπό έχει να λειτουργήσει ως ένα βοηθητικό εργαλείο, ώστε:

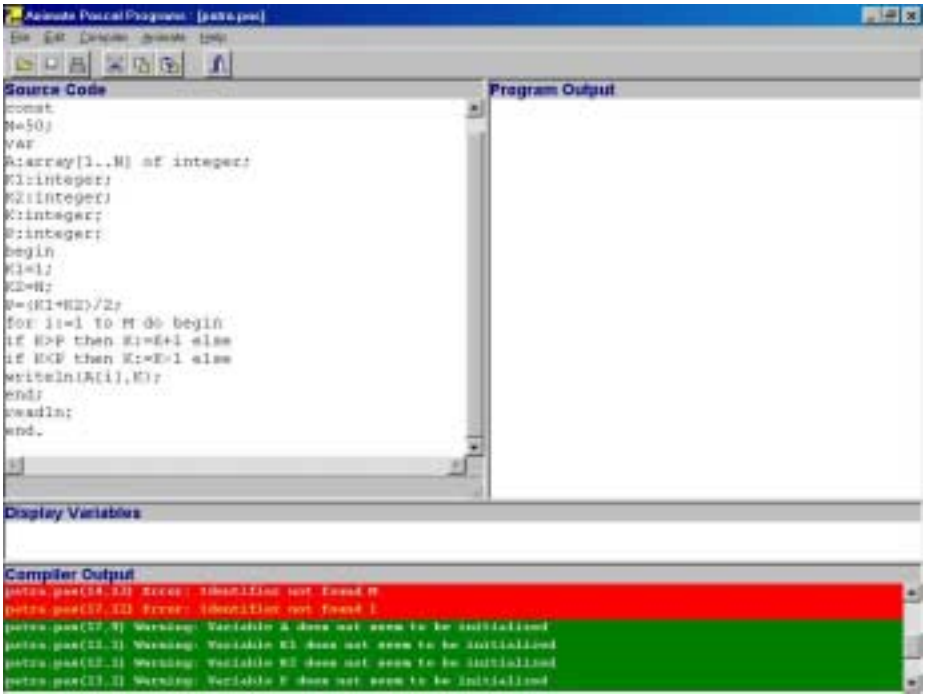
- οι σπουδαστές να αντιμετωπίσουν τα προβλήματα που συναντούν στην κατανόηση και εφαρμογή των βασικών εννοιών του δομημένου προγραμματισμού,
- να αποδώσει τη δυναμική φύση της εκτέλεσης προγραμμάτων,
- οι καθηγητές να έχουν τη δυνατότητα εντοπισμού των λανθασμένων αντιλήψεων των σπουδαστών.

[1] Το AnimPascal σχεδιάστηκε και αναπτύχθηκε στο έργο «Αναμόρφωση του Προγράμματος Σπουδών με την Εισαγωγή Παραστατικών Μεθόδων» του τμήματος Εφαρμοσμένης Πληροφορικής στα πλαίσια του ΕΠΕΑΕΚ

Οι κύριες λειτουργίες που προσφέρει το εκπαιδευτικό λογισμικό για την εκπλήρωση των παραπάνω στόχων είναι:

- Η δυνατότητα κωδικοποίησης ενός προγράμματος στη γλώσσα προγραμματισμού Pascal και η μεταγλώττιση του.
- Η δυναμική οπτικοποίηση της εκτέλεσης του προγράμματος.
- Η καταγραφή των προγραμμάτων του χρήστη και των αντίστοιχων αποτελεσμάτων της μεταγλώττισης.

Η απλότητα, σαφήνεια και ευελιξία του Γραφικού Συστήματος Επικοινωνίας με το Χρήστη (Graphical User Interface) της εκπαιδευτικής εφαρμογής συνεισφέρουν αποτελεσματικά στην πραγματοποίηση των δυνατοτήτων της. Το κύριο παράθυρο του AnimPascal αποτελείται από έξι δομικά συστατικά: το *μενού*, τη *μπάρα εργαλείων* και τέσσερις *περιοχές*, όπως φαίνεται στην Εικόνα 1.



Εικόνα 1: Το κύριο παράθυρο του AnimPascal

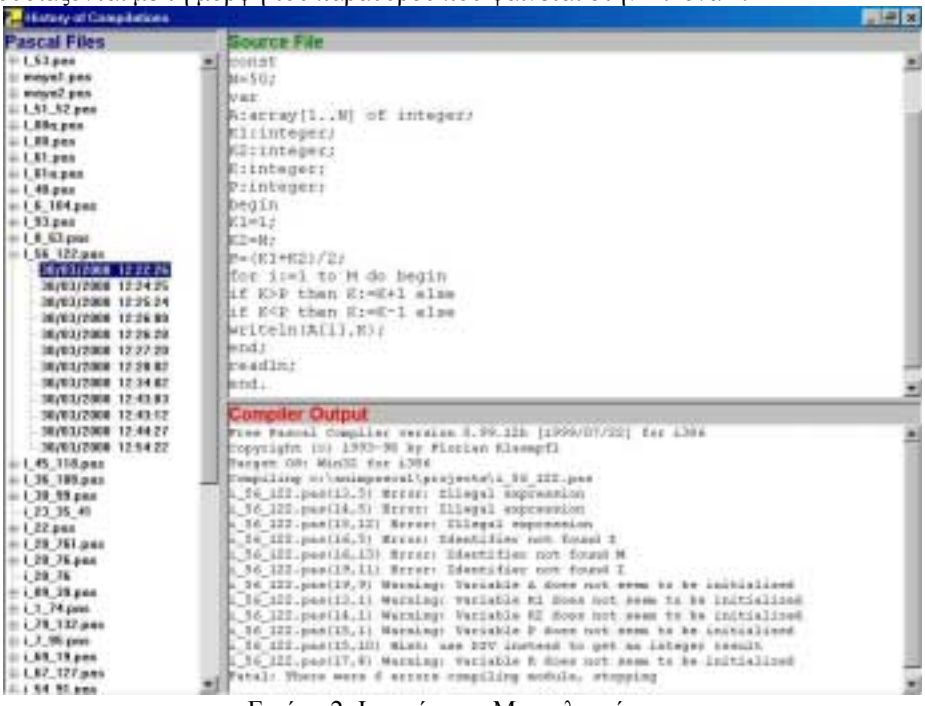
Στη συνέχεια θα παρουσιαστούν οι λειτουργίες του μενού της εφαρμογής και των τεσσάρων περιοχών. Δε θα δοθεί έμφαση στη μπάρα εργαλείων γιατί οι επιλογές της έχουν ως αποτέλεσμα τη γρηγορότερη προσπέλαση των συνήθων επιλογών του μενού. Στην *περιοχή Source Code (Κώδικας Προγράμματος*), ο χρήστης έχει τη δυνατότητα να γράψει, τροποποιήσει, αποθηκεύσει ή ανακτήσει ένα ήδη αποθηκευμένο πρόγραμμα στη γλώσσα προγραμματισμού Pascal. Οι λειτουργίες αυτές πραγματοποιούνται με τις επιλογές των μενού *File* και *Edit*.

Η *περιοχή Compiler Output (Έξοδος Μεταγλωττιστή*) παρουσιάζει τα αποτελέσματα της μεταγλώττισης του προγράμματος της *περιοχής Source Code*, όπως φαίνεται στην Εικόνα 1. Ενδιαφέρον παρουσιάζει ο τρόπος που αντιμετωπίζεται με τη βοήθεια του AnimPascal η περίπτωση κατά την οποία, ο μεταγλωττιστής ενημερώνει το χρήστη για σφάλματα (errors) ή τον προειδοποιεί για πιθανές παραλείψεις (warnings, hints, notes) του προγράμματος του. Στην πρώτη περίπτωση το υπόβαθρο της αντίστοιχης γραμμής στην *περιοχή Compiler Output* χρωματίζεται με κόκκινο, ενώ στη δεύτερη με πράσινο χρώμα. Και στις δύο περιπτώσεις το χρώμα των γραμμάτων αλλάζει από μαύρο σε άσπρο. Οι αλλαγές στην εμφάνιση των αποτελεσμάτων της μεταγλώττισης έχουν δυναμικό χαρακτήρα. Ο σπουδαστής μπορεί να

αλληλεπιδράσει με αυτές: Αν επιλέξει μία κόκκινη ή πράσινη γραμμή τότε ο δρομέας της *περιοχής Source Code* μεταφέρεται στην γραμμή και στήλη του προγράμματος που αντιστοιχεί το μήνυμα του μεταγλωττιστή. Γίνεται φανερό ότι η *περιοχή Compiler Output* προσφέρει μία δυναμική οπτικοποίηση των αποτελεσμάτων της μεταγλώττισης, την οποία δεν προσφέρουν τα συνήθη ολοκληρωμένα περιβάλλοντα προγραμματισμού. Τα οφέλη που προκύπτουν είναι σημαντικά για τον αρχάριο προγραμματιστή:

- Προσελκύει την προσοχή και το ενδιαφέρον του.
- Ο χρόνος που απαιτείται για την ανάπτυξη ενός προγράμματος κατανέμεται στα ουσιαστικά σημεία της διαδικασίας του προγραμματισμού και όχι σε χρονοβόρες και επίπονες διαδικασίες, όπως η εύρεση της λαθεμένης γραμμής του κώδικα.
- Γίνεται φανερή η σημασία των υποδείξεων του μεταγλωττιστή για την κωδικοποίηση βελτιστοποιημένων προγραμμάτων και την αποφυγή των σφαλμάτων, που συνήθως προκύπτουν, όταν οι προγραμματιστές δεν τις λαμβάνουν υπόψη τους.

Η μεταγλώττιση του υπό επεξεργασία προγράμματος πραγματοποιείται με την επιλογή *Compile* του μενού *Compiler*. Η παραπάνω ενέργεια δίνει το επιπρόσθετο αποτέλεσμα της καταγραφής των ενεργειών των χρηστών. Το AnimPascal αποθηκεύει το πρόγραμμα, τα αποτελέσματα της μεταγλώττισης, την ημερομηνία και ώρα που πραγματοποιήθηκε η τελευταία. Με την επιλογή *History* του μενού *Compile*, οι παραπάνω πληροφορίες παρουσιάζονται με τη μορφή του παράθυρου που φαίνεται στην Εικόνα 2.



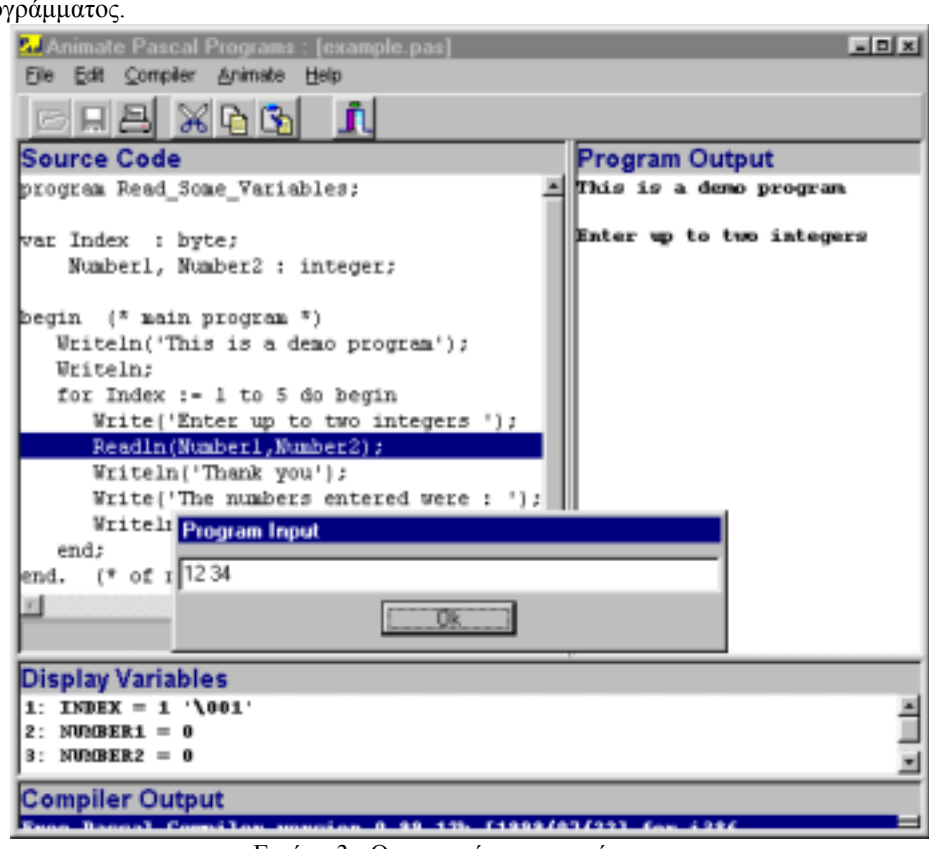
Εικόνα 2: Ιστορία των Μεταγλωττίσεων

Το *παράθυρο History of Compilations (Ιστορία Μεταγλωττίσεων*) διαιρείται σε τρεις περιοχές, την *περιοχή Pascal Files (Αρχεία Pascal*), η οποία παρουσιάζει με τη μορφή δένδρου δυο επιπέδων τις ενέργειες του χρήστη. Στο πρώτο επίπεδο, εμφανίζονται τα ονόματα των αρχείων που επεξεργάστηκε ο χρήστης. Το δεύτερο περιλαμβάνει τις ημερομηνίες και τις ώρες της επεξεργασίας. Όταν επιλεγεί μία από αυτές (του 2ου επιπέδου), στην *περιοχή Source File* παρουσιάζεται το αντίστοιχο πρόγραμμα και στην *περιοχή Compiler Output* τα αποτελέσματα της μεταγλώττισης του. Ο σκοπός του παράθυρου *History of Compilations* είναι να βοηθήσει τον καθηγητή να καταγράψει τα συχνότερα λάθη και τις αντιλήψεις των σπουδαστών της τάξης ώστε να προσαρμόσει κατάλληλα τις διαλέξεις του.

Οι επιλογές του μενού *Animate* παρέχουν τη δυνατότητα της δυναμικής οπτικοποίησης της εκτέλεσης του προγράμματος, είτε μέσω της επιλογής *Trace To Next Source Line* (*εντολή προς εντολή*) είτε μέσω της επιλογής *Animate To Cursor* δηλαδή οπτικοποίηση μέχρι το σημείο που υποδεικνύει ο δείκτης του ποντικιού. Σε κάθε περίπτωση ο τρόπος οπτικοποίησης του προγράμματος είναι ο εξής: Τονίζεται η γραμμή του κώδικα που θα εκτελεστεί, με αλλαγή του χρώματος των χαρακτήρων και του υποβάθρου της. Η εκτέλεση της τρέχουσας εντολής, όπως φαίνεται στην Εικόνα 3:

- όταν αναφέρεται σε απόδοση τιμής μιας μεταβλητής, οδηγεί στην ενημέρωση της *περιοχής Display Variables (Εμφάνισης Μεταβλητών Προγράμματος*) με τις μεταβλητές και τις αντίστοιχες τιμές τους. Η *περιοχή Display Variables* αποτελεί το συμβολικό χώρο οπτικοποίησης της μνήμης που καταλαμβάνει το πρόγραμμα.
- όταν είναι εντολή εξόδου, οδηγεί στη προσθήκη των αποτελεσμάτων της στην *περιοχή Program Output*.
- όταν είναι εντολή εισόδου, τότε εμφανίζεται ένα παράθυρο με τίτλο *Program Input* και ο χρήστης καλείται να δώσει τις αντίστοιχες τιμές στο πρόγραμμα.

Η δυναμική οπτικοποίηση της εκτέλεσης του προγράμματος, εντολή προς εντολή, δίνει τον απαιτούμενο χρόνο στο σπουδαστή για την κατανόηση του τρόπου λειτουργίας της τρέχουσας εντολής. Από την άλλη πλευρά, η οπτικοποίηση του προγράμματος μέχρι τη γραμμή που υποδεικνύει ο δείκτης του ποντικιού, έχει ως αποτέλεσμα το γρήγορο πέρασμα των εντολών που έχει κατανοήσει και την εστίαση της προσοχής του στο επιθυμητό σημείο του προγράμματος.



Εικόνα 3 : Οπτικοποίηση προγράμματος

Το AnimPascal δίνει τη δυνατότητα του ελέγχου του χρόνου που απαιτείται για την εκτέλεση μιας εντολής μέσα από τις αντίστοιχες επιλογές του μενού *Animate*. Ο σπουδαστής μπορεί να

προσαρμόσει τον αντίστοιχο χρόνο στις ανάγκες του και ο καθηγητής στις ανάγκες της διδασκαλίας στην τάξη.
Η οπτικοποίηση της εκτέλεσης του προγράμματος μπορεί να σταματήσει με την επιλογή *Terminate* του μενού *Animate*, όποτε κρίνεται αναγκαίο.
Το λογισμικό AnimPascal έχει γραφτεί με τη γλώσσα προγραμματισμού C++ και εκτελείται στην πλατφόρμα των Windows 95/98. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ο μεταγλωττιστής Free Pascal [4] και ο αποσφαλματωτής gdb [9], αφού ο συγκεκριμένος μεταγλωττιστής υποστηρίζει την αντίστοιχη πληροφορία αποσφαλμάτωσης. Τα λογισμικά αυτά είναι διαθέσιμα κάτω από τη GNU Public License. Ο Free Pascal χρησιμοποιήθηκε λόγω της σημασιολογικής του συμβατότητας με τον Turbo Pascal, τον πιο συνήθη μεταγλωττιστή για τη διδασκαλία της γλώσσας προγραμματισμού Pascal. Επιπρόσθετα, τα αποτελέσματα της μεταγλώττισης είναι πιο εκτεταμένα και περιέχουν υποδείξεις χρήσιμες τόσο για τον αρχάριο όσο και τον έμπειρο προγραμματιστή. Είναι διαθέσιμος για διαφορετικούς επεξεργαστές και λειτουργικά συστήματα, πράγμα που δίνει τη δυνατότητα δημιουργίας εκδόσεων του AnimPascal και σε διαφορετικές πλατφόρμες.

3. Η Χρήση του AnimPascal

Το περιβάλλον AnimPascal χρησιμοποιήθηκε πειραματικά σε φοιτητές του Α΄ έτους τμήματος Πληροφορικής, στα πλαίσια του εισαγωγικού μαθήματος στον προγραμματισμό. Το μάθημα περιελάμβανε 2 ώρες διάλεξη και 2 ώρες εργαστηριακό μάθημα με την επίβλεψη του διδάσκοντα. Στο εργαστηριακό μάθημα οι φοιτητές χρησιμοποιούσαν το AnimPascal. Στα πλαίσια του μαθήματος προτάθηκε στους φοιτητές να λύσουν το πρόβλημα της δυαδικής αναζήτησης (Binary search). Το πρόβλημα διατυπώθηκε με τον εξής τρόπο:
Δίνεται ένας πίνακας Α με Μ στοιχεία ταξινομημένα σε αύξουσα διάταξη και ένας αριθμός, έστω Κ, και ζητάμε να γίνει πρόγραμμα που θα εντοπίζει τον αριθμό Κ, αν υπάρχει μέσα στον δοσμένο πίνακα. Αν υπάρχει θα εμφανίζει τη θέση του Κ μέσα στον πίνακα αλλιώς θα εμφανίζει τον αριθμό 0. Ζητείται να υλοποιηθεί με τη μέθοδο της Δυαδικής Αναζήτησης (Binary Search).
Ο αλγόριθμος δυαδικής αναζήτησης προτάθηκε γιατί είναι γνωστός στη επιστημονική κοινότητα για την φαινομενική του απλότητα. Ενώ ο αλγόριθμος μοιάζει να είναι απλός, στην πραγματικότητα η ορθή κατασκευή του και η πιστοποίηση της ορθότητας του [1] παρουσιάζουν ορισμένες ιδιαιτερότητες. Ο R. Lesuisse [7] εξάλλου έδειξε ότι ακόμη και ορισμένοι δημοσιευμένοι αλγόριθμοι για το πρόβλημα της δυαδικής αναζήτησης περιέχουν λάθη, αδυναμίες, ιδιαίτερες συνθήκες (όπως για παράδειγμα τα στοιχεία του αρχικού πίνακα να είναι 2^n σε πλήθος).
Ο αλγόριθμος της δυαδικής αναζήτησης είχε διδαχθεί στους φοιτητές ενάμισι μήνα περίπου πριν την εφαρμογή του στο εργαστηριακό μάθημα. Ο στόχος του εργαστηριακού αυτού μαθήματος ήταν

- η καταγραφή της επίδρασης του προγραμματιστικού περιβάλλοντος στους φοιτητές,
- η καταγραφή των αντιλήψεων των φοιτητών και η επιβεβαίωση των προβλημάτων που έχουν εντοπιστεί σχετικά με το πρόβλημα της δυαδικής αναζήτησης
- η παρακολούθηση, όσο ήταν δυνατόν, της πορείας που ακολουθούν οι φοιτητές στη φάση της ανάπτυξης, του ελέγχου και της αποσφαλμάτωσης ενός προγράμματος για το οποίο έχει διαπιστωθεί από διάφορους ερευνητές ότι παρουσιάζει δυσκολίες.

Από τα προγράμματα των σπουδαστών μπορούμε να κάνουμε τις εξής παρατηρήσεις:

- Η δυνατότητα του μεταγλωττιστή του AnimPascal να εμφανίζει warnings και hints βοήθησε τους φοιτητές στην κατανόηση της πραγματικής σημασίας ορισμένων εντολών. Ως παράδειγμα μπορούμε να αναφέρουμε το εξής: από την μελέτη των διάφορων διαδοχικών εκδόσεων των προγραμμάτων τους διαπιστώσαμε ότι ενώ στην αρχή χρησιμοποιούσουν την σχέση (1) για τον υπολογισμό της θέσης του μεσαίου στοιχείου του πίνακα, γρήγορα έκαναν χρήση του τελεστή div (ένα από τα hints που πρότεινε ο μεταγλωττιστής) και έτσι διόρθωναν το συντακτικό λάθος που είχε σχέση με την ασυμβατότητα τύπων που δεν υποστηρίζει η Pascal.

```
Midpoint := (LowLimit+HighLimit) / 2                (1)
Midpoint := (LowLimit+HighLimit) div 2              (2)
```

Αυτό είχε ως αποτέλεσμα να επικεντρώσουν τις προσπάθειές τους στην ανάπτυξη του προγράμματος και στον έλεγχο της ορθότητας του, ενώ άλλοι σπουδαστές προσπαθούσαν να τροποποιήσουν τη σχέση (1) πιστεύοντας, λόγω των μηνυμάτων λάθους, ότι είχε λογικό λάθος.

- Ένας σημαντικός αριθμός σπουδαστών έκανε λογικά λάθη, συγχέοντας τη θέση στοιχείου πίνακα με το στοιχείο του πίνακα, δηλαδή το Ι με το Α[Ι] ή ακόμη υπολογίζοντας κάθε φορά λανθασμένα τα όρια του πίνακα μέσα στο οποίο πιθανόν να εντοπίζεται το αναζητούμενο στοιχείο - φαινόμενο συχνό κατά τη διδασκαλία και χρήση των πινάκων.
- Ενας σημαντικός αριθμός σπουδαστών δεν έλαβε υπόψη του τις «ακραίες» περιπτώσεις (για τους φοιτητές), όπως: ο αρχικός πίνακας να μην έχει στοιχεία ή να έχει μόνον ένα στοιχείο ή ακόμη και το αναζητούμενο στοιχείο να μην είναι στοιχείο του πίνακα.
- Αντίθετα υπήρξε και ένας αριθμός φοιτητών που προσπάθησαν να ελέγξουν την ορθότητα της λύσης που διατύπωσαν (αυτό φαίνεται από την παρακολούθηση της ιστορίας των μεταγλωττίσεων, όταν πετύχαιναν μια έκδοση χωρίς συντακτικά λάθη). Στη προσπάθεια τους αυτή, συνήθως, τροποποιούσαν τη συνθήκη τερματισμού προκειμένου να αποφύγουν περιπτώσεις ατέρμονης εκτέλεσης. Στις περιπτώσεις αυτές πρότειναν ως συνθήκη τερματισμού αρκετά διαφορετική απ' αυτή που είχαν διδαχθεί στη διάλεξη, και εκεί γίνεται φανερό ότι προσπαθούσαν να ελέγξουν την ορθότητα του προγράμματός τους για διάφορες περιπτώσεις που πίστευαν ότι μπορούσαν να συμβούν.
- Ο κύριος όγκος των σπουδαστών ωστόσο κατασκεύασε αλγόριθμους οι οποίοι «έπεφταν» σε infinity loops. Ο κύριος λόγος ήταν ότι οι σπουδαστές αντικαθιστούσαν τα όρια με το ενδιάμεσο στοιχείο

```
LowLimit:=1;
HighLimit:=n;
…..
Midpoint:=(LowLimit+HighLimit) div 2;
If v <> A[Midpoint] then
    if v > A[Midpoint] then LowLimit:= Midpoint else
HighLimit:= Midpoint;
```

κάνοντας την (έμμεση) υπόθεση ότι το διάστημα αναζήτησης συνεχώς μικραίνει. Το γεγονός ωστόσο είναι ότι αυτό δεν ισχύει πάντοτε – για παράδειγμα αν HighLimit-LowLimit=1. Έτσι και στην περίπτωση αυτή συμπεραίνουμε ότι ο αλγόριθμος πιθανόν να μην ελέγχονταν συστηματικά και διεξοδικά – και απλώς οι σπουδαστές να στηρίζονταν στη διαίσθησή τους.

- Ο R. Lesuisse παρατήρησε ότι μια κατηγορία αλγορίθμων παρουσιάζουν τμήματα που επαναλαμβάνονται – δείγμα «φτωχής» προγραμματιστικής ανάλυσης. Έκανε την υπόθεση ότι τα προγράμματα αυτά αντιστοιχούν σε κατασκευές που πραγματοποιήθηκαν αποκλειστικά με *νοητή προσομοίωση της εκτέλεσης*. Αλγόριθμους αυτού του τύπου συναντήσαμε κι εμείς – και θεωρούμε την υπόθεση του R. Lesuisse πολύ ισχυρή. Η επανάληψη των κομματιών δείχνει ότι οι σπουδαστές μετά την διαπίστωση του τμήματος στο οποίο πρέπει να ευρίσκεται το αναζητούμενο στοιχείο, προσπαθούν και πάλι να διχοτομήσουν το νέο «υποδιάστημα» του πίνακα.

Στη συνέχεια παραθέτουμε τον πίνακα 1 που παρουσιάζει αναλυτικά το τι παρατηρήθηκε στο πείραμα.

1	Δεν έκαναν σχεδόν τίποτα	1,4%
2	Το έλυσαν σωστά	5,8%
3	Η Τελική έκδοση του προγράμματος είχε συντακτικά λάθη	4,3%
4	Χώριζαν σε 2 τμήματα τον αρχικό πίνακα και μετά έκαναν σειριακό ψάξιμο σε καθένα από τα 2 τμήματα	4,3%

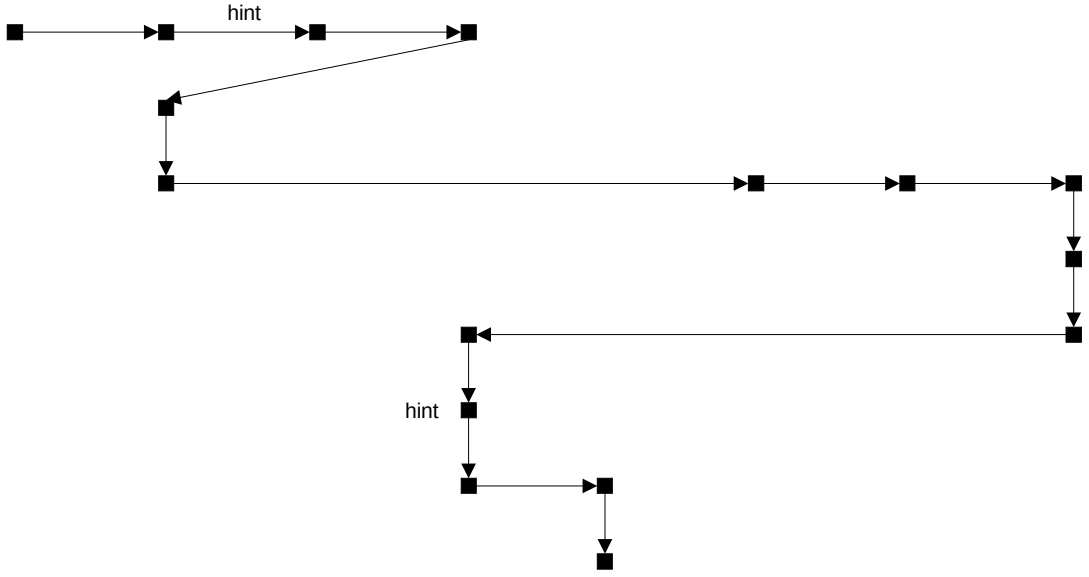
5	Δεν χρησιμοποίησαν επαναληπτική δομή	4,3%
6	Λάθος συνθήκη τερματισμού (συσχετίζεται με τη παρατήρηση 9)	17,4%
7	Υπολόγιζαν μια φορά τη θέση του μεσαίου στοιχείου και στη συνέχεια δεν την υπολόγιζαν για κάθε τρέχοντα υποπίνακα	1,4%
8	Θεωρούσαν ότι το αναζητούμενο στοιχείο υπάρχει οπωσδήποτε στον πίνακα	8,7%
9	Λάθος υπολογισμός της θέσης του μεσαίου στοιχείου	7,2%
10	Λάθος υπολογισμός των νέων ορίων των υποπινάκων	10,1%
11	Συγχέουν τη θέση στοιχείου πίνακα με το στοιχείο	10,1%
12	Λάθος συνθήκη εμφάνισης αποτελεσμάτων ενώ είχαν σωστή τη συνθήκη τερματισμού	1,4%
13	Χρήση for και υπολόγιζαν εκ νέου τις τιμές των δεικτών του for πιστεύοντας ότι έτσι μειώνεται το μέγεθος του πίνακα	1,4%
14	Προβλήματα στη διατύπωση της συνθήκης τερματισμού (χρήση του τελεστή AND) πράγμα που προκάλεσε συνεχείς αλλαγές του κώδικα	2,9%

Πίνακας 1: Συγκεντρωτικός πίνακας σχετικά με τα λάθη των σπουδαστών

Ο πίνακας 1 προέκυψε από μια συστηματική καταγραφή των λαθών των σπουδαστών από την εξέταση των διαδοχικών εκδόσεων των προγραμμάτων τους. Ωστόσο η καταγραφή αυτή δίνει στην πραγματικότητα πολύ πιο ισχυρά στοιχεία ποιοτικού χαρακτήρα για κάθε σπουδαστή.

Ο πίνακας 2 προυσιάζει μια τυπική περίπτωση πορείας ενός σπουδαστή και η *χρονογραμμή* (time line) της πορείας αυτής – ιδιαίτερα αν εμπλουτιστεί με στοιχεία χρόνου – μπορεί να δώσει μια πληρέστερη εικόνα για την διαδρομή της επίλυσης του προβλήματος.

ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ			**ΛΟΓΙΚΑ ΛΑΘΗ**		**ΣΗΜΑΣΙΟΛΟΓΙΚΑ ΛΑΘΗ**		
Παρένθεση αντί για αγκύλη	Χρήση div	Λανθασμένη χρήση του if…then…else	While και and	Απόδοση τιμής	Αρχικοποίηση LowLimit, HihgLimit	Σύγχυση I, A[I]	Συνθήκη while



Πίνακας 2: Τυπική περίπτωση πορείας ενός σπουδαστή και η *χρονογραμμή* της πορείας αυτής

Ο πίνακας που συγκεντρώνει τα λάθη ενός σπουδαστή καθώς και η αντίστοιχη χρονογραμμή γίνεται από τον διδάσκοντα μελετώντας τις διαδοχικές εκδόσεις του προγράμματος του

σπουδαστή και δεν προσφέρονται αυτοματοποιημένα από την παρούσα έκδοση του λογισμικού.

Από τον παραπάνω ενδεικτικά παρουσιαζόμενη χρονογραμμή μπορούμε να διαπιστώσουμε τα εξής:

- Η συνεχής παλινδρόμηση ανάμεσα στα συντακτικά και λογικά λάθη δείχνει με σαφή τρόπο τις δυσκολίες που παρουσιάζει για τον αρχάριο σπουδαστή ο ιδιοσυγκρασιακός χαρακτήρας των «πραγματικών» γλωσσών προγραμματισμού. Ιδιαίτερα οι επαναλαμβανόμενες απόπειρες διόρθωσης των λαθών που έχουν συντακτική προέλευση, δείχνει με αδιαμφισβήτητο τρόπο ότι η αρχική επαφή των σπουδαστών με τον προγραμματισμό πρέπει να γίνεται στα πλαίσια γλωσσών χωρίς ιδιαιτερότητες στη σύνταξή τους. Στον πίνακα δεν είναι εμφανές, αλλά αν συνυπολογιστεί ο χρόνος που αφιερώνεται στα συντακτικά λάθη, καθίσταται φανερό ότι η σύνταξη της χρησιμοποιούμενης γλώσσας αποτελεί έναν ισχυρό αρνητικό παράγοντα στην επίλυση του προβλήματος.
- Με έναν τελείως ανάλογο τρόπο, καθίσταται φανερό ότι τα μηνύματα του συστήματος προς το σπουδαστή παίζουν σπουδαίο ρόλο στη διαδικασία επίλυσης του προβλήματος. Έτσι, στο παράδειγμά μας, η επιτυχημένη υπόδειξη της χρήσης του τελεστή DIV στη θέση της διαίρεσης αποτελεί μια ουσιαστική βοήθεια προς τον χρήστη, ενώ το μήνυμα για τα όρια του πίνακα δε γίνεται αντιληπτό από το σπουδαστή ο οποίος κάνει επανειλημμένες απόπειρες για τη διόρθωση των λαθών του, χωρίς να είναι σε θέση να ερμηνεύσει το νόημα του μηνύματος του μεταγλωττιστή. Στο σημείο αυτό θα μπορούσαμε να πούμε ότι η συστηματική μελέτη των λαθών των σπουδαστών και διερεύνηση των λόγων για τους οποίους οι σπουδαστές δεν δείχνουν να μπορούν να αξιοποιήσουν αποτελεσματικά τα μηνύματα του μεταγλωττιστή θα βοηθούσε στο να αυξήσουμε την αποτελεσματικότητα των μηνυμάτων αυτών. Από τη μέχρι τώρα εμπειρική μελέτη που έχουμε διενεργήσει στο θέμα αυτό, μπορούμε να πούμε ότι ο εξελληνισμός των μηνυμάτων και η εκτενέστερη περιγραφή του κάθε μηνύματος θα μπορούσε να αυξήσει την αποτελεσματικότητα του.
- Η μελέτη των χρονογραμμών δείχνει επίσης με σαφή τρόπο τα σημεία του αλγορίθμου στα οποία οι σπουδαστές συνάντησαν μεγάλες δυσκολίες – όπως για παράδειγμα το θέμα των δυο ορίων της επόμενης περιοχής αναζήτησης: άσχετα από το τελικό αποτέλεσμα, όλοι σχεδόν οι σπουδαστές έκαναν λάθη στο σημείο αυτό. Οι χρονογραμμές επίσης υποδεικνύουν έμμεσα τους τρόπους σκέψης των σπουδαστών. Είναι, για παράδειγμα, σαφές ότι η πλειοψηφία των σπουδαστών κατασκευάζει τα προγράμματά της χρησιμοποιώντας την προσομοίωση και ένα είδος νοητικής εκτέλεσης του προγράμματος.

4. Συμπεράσματα

Η μέχρι τώρα χρήση του AnimPascal έδειξε ότι μπορεί να βοηθήσει σημαντικά τον αρχάριο στην απόκτηση της ικανότητας καθορισμού, ανάπτυξης, ελέγχου και αποσφαλμάτωσης ενός προγράμματος. Ακόμη μπορεί να προσφέρει πληροφορίες στον διδάσκοντα σχετικά με τη πορεία που ακολουθούν οι σπουδαστές στην ανάπτυξη, έλεγχο και αποσφαλμάτωση ενός προγράμματος.

Συνοψίζοντας, τονίζουμε τα βασικά πλεονεκτήματα και χρήσεις του AnimPascal καθώς και τα στοιχεία που θα προστεθούν σε μελλοντικές εκδόσεις του.

Οφέλη

- Δυναμική οπτικοποίηση αποτελεσμάτων μεταγλώττισης και της εκτέλεσης προγραμμάτων σε Pascal.
- Καλύτερη κατανομή χρόνου στη μελέτη της εκτέλεσης ενός προγράμματος και στη διδασκαλία των εισαγωγικών εννοιών του προγραμματισμού.
- Χρήση στα εργαστήρια που συνήθως συνοδεύουν τα εισαγωγικά μαθήματα στον προγραμματισμό.
- Δυνατότητα επανάληψης ενός μαθήματος στο προσωπικό χώρο μελέτης του σπουδαστή.

- Κινητοποίηση και προσέλκυση του ενδιαφέροντος του σπουδαστή και δημιουργία κινήτρου για τον προγραμματισμό.
- Κατασκευή προγραμμάτων από το σπουδαστή και δημιουργία γνώσης από τον ίδιο για τον τρόπο που εκτελούνται τα προγράμματα.
- Μελέτη και ταξινόμηση των αντιλήψεων των σπουδαστών για τον τρόπο που επιλύουν ένα πρόβλημα με τη χρήση της γλώσσας προγραμματισμού Pascal.
- Χρήσιμο εργαλείο για τον διδάσκοντα στη φάση της διδασκαλίας ενός έτοιμου προγράμματος – παραδείγματος, αφού με τη βοήθεια του AnimPascal μπορεί να δείξει με λεπτομέρεια καθετί που αφορά στην εκτέλεση του προγράμματος.

Προσθήκες-Μειονεκτήματα

Σχετικά με το γραφικό ενδιάμεσο οι βελτιώσεις που θα μπορούσαν να πραγματοποιηθούν είναι:

- Η οπτικοποίηση των υποπρογραμμάτων αφού στην παρούσα πιλοτική έκδοση της εκπαιδευτικής εφαρμογής οπτικοποιείται μόνο το κύριο μέρος ενός προγράμματος Pascal.
- Η οπτικοποίηση της εκτέλεσης προς τα πίσω ενός προγράμματος θα προσφέρει πολλά εκπαιδευτικά οφέλη.
- Ο τονισμός των γραμμών κώδικα θα μπορούσε να γίνει τμηματικά ώστε οι σπουδαστές να αντιλαμβάνονται καλύτερα τις περιπτώσεις των δομών ελέγχου, επιλογής, επανάληψης κ.ο.κ.

Σχετικά με την παρουσίαση των ενεργειών των σπουδαστών και την περαιτέρω αξιοποίησή τους:

Το AnimPascal στη σημερινή του έκδοση καταγράφει και παρουσιάζει τις διάφορες εκδόσεις των προγραμμάτων των σπουδαστών και τα αντίστοιχα αποτελέσματα της μεταγλώττισής τους. Η αυτοματοποιημένη αξιοποίηση των καταχωρήσεων είναι ένα ιδιαίτερα ενδιαφέρον θέμα και σχεδιάζεται να ενσωματωθεί σε μελλοντική έκδοση του λογισμικού. Το μοντέλο, που σχεδιάζεται με σκοπό την ενσωμάτωση του στην εκπαιδευτική εφαρμογή, αποτελείται από δύο διακεκριμένες διαδικασίες, οι οποίες παρουσιάζονται και αναλύονται παρακάτω:

- *Ταξινόμηση των μηνυμάτων μεταγλώττισης*: Η ταξινόμηση των μηνυμάτων του μεταγλωττιστή είναι δυνατόν να αυτοματοποιηθεί εύκολα, με σκοπό την εφαρμογή στατιστικών μεθόδων για την εξαγωγή συμπερασμάτων που αφορούν στα συχνότερα λάθη των σπουδαστών, στην κατηγορία στην οποία ανήκουν και στο χρόνο που αφιερώνουν για την αντιμετώπιση τους, όταν τα λαμβάνουν υπόψη τους.
- *Παρουσίαση διαφορών μεταξύ δύο διαδοχικών ενεργειών για την επίλυση του προβλήματος*: Είναι σημαντική η εποπτική παρουσίαση των διαφορών μεταξύ δύο διαδοχικών εκδόσεων των προγραμμάτων των σπουδαστών και των αντίστοιχων διαφορών των αποτελεσμάτων του μεταγλωττιστή. Η δυνατότητα αυτή μπορεί να πραγματοποιηθεί με τη χρήση εργαλείων, όπως το diff [11]. Με αυτό τον τρόπο είναι δυνατό να εξαχθούν συμπεράσματα για το βαθμό κατανόησης των μηνυμάτων του μεταγλωττιστή, βοήθειας που προσφέρουν στην επίλυση του προβλήματος και τις αντιλήψεις των σπουδαστών γι αυτά. Επιπρόσθετα, όταν τα προγράμματα είναι απαλλαγμένα από συντακτικά λάθη ή μηνύματα που μπορούν να οδηγήσουν σε σφάλματα, τότε μπορεί να γίνει φανερός ο τρόπος αντιμετώπισης του δεδομένου προβλήματος από ένα σπουδαστή.

Βιβλιογραφία

1. Bentley J., (1986), Programming Pearls, Addison-Wesley,.
2. 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, In Twenty-sixth SICGSE Technical Symposium on Computer Science Education (*SICGSE Bulletin*) vol. 27, pp. 29-33.

3. Brusilovski, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller P., (1997), Mini-languages: a way to learn programming principles, *Education and Information Technologies 2*, pp. 65-83.
4. Canneyt M. V., Klämpfl F., (1999), Free Pascal: Users' Manual. Είναι διαθέσιμο στο δικτυακό τόπο http://www.brain.uni-freiburg.de/~klaus/fpc/docs.html.
5. Dagdilelis V., Satratzemi M., (1998), DIDAGRAPH: A Software for Teaching Graph Theory Algorithms, *SIGSE Bulletin, ACM Press*,30 (3), pp. 64-68.
6. du Boulay, B., (1989), Some Difficulties Of Learning To Program, *In Studying The Novice Programmer*, Soloway, E., Sprohrer, J. (Eds.) Lawrence Erlbaum Associates, pp. 283-300.
7. Lesuisse R., Some Lessons Drawn from the History of the Binary Search Algorithm, *The Computer Journal*, Vol. 26, n. 2, 1983, pp. 154-163.
8. 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, pp. 3-28.
9. Stallman, R., and Cygnus Support (1995), Debugging with GDB, Free Software Foundation, Boston.
10. 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.
11. Diff, είναι διαθέσιμο στο δικτυακό τόπο http://www.cslab.vt.edu/manuals/diff/diff_toc.html.