Υπερβαίνοντας τα στερεότυπα του δομημένου προγραμματισμού

Ευριπίδης Βραχνός

Ζάννειο Πειραματικό Γυμνάσιο / Πανεπιστήμιο Δυτικής Αττικής
evrachnos@{gmail.com, uniwa.gr}

Περίληψη

Η εντολή άλματος goto έχει εδώ και πολλά χρόνια τεθεί στο περιθώριο τόσο σε εκπαιδευτικό όσο και σε επαγγελματικό επίπεδο, ενώ άλλες εντολές διακλάδωσης όπως η break και η return χρησιμοποιούνται σήμερα ευρέως στην ανάπτυξη εφαρμογών αλλά και στην εκπαίδευση. Παρόλα αυτά η χρήση αυτών των εντολών δεν είναι ακόμα αποδεκτή από την εκπαιδευτική κοινότητα. Το βασικό ερώτημα είναι αν οι εντολές αυτές έχουν να προσφέρουν κάτι στην διδασκαλία του προγραμματισμού και της αλγοριθμικής ή αν πράγματι δημιουργούν τα προβλήματα για τα οποία τις κατηγορούν. Στο πλαίσιο αυτό διενεργήσαμε μια έρευνα πεδίου σε 40 πρωτοετείς φοιτητές και 14 μαθητές γυμνασίου με σκοπό τη διερεύνηση των δυσκολιών που αντιμετωπίζουν οι φοιτητές με τις εντολές αυτές. Ερευνήσαμε αν οι φοιτητές αποφεύγουν τη χρήση αυτών των εντολών ή τις προτιμούν έναντι δομημένων αλγορίθμων. Τα συμπεράσματα της έρευνας συμφωνούν με τα αποτελέσματα όλων των αντίστοιχων ερευνών και δείχνουν την χρησιμότητα των εντολών διακλάδωσης σε ένα εισαγωγικό μάθημα προγραμματισμού. Ωστόσο εντοπίστηκε μια σημαντική παρανόηση που έχουν οι φοιτητές όταν χρησιμοποιούν την εντολή break εντός ένθετων εντολών επανάληψης, η οποία δεν έχει αναφερθεί μέχρι στιγμής στην βιβλιογραφία.

Λέξεις κλειδιά: εντολές διακλάδωσης, break, return, python, δομημένος προγραμματισμός

1. Εισαγωγή

Ο βασικός στόχος ενός εισαγωγικού μαθήματος προγραμματισμού είναι να διδάξει στους μαθητές τις βασικές προγραμματιστικές δομές και μεθοδολογίες, τις οποίες θα χρησιμοποιήσουν για την αλγοριθμική επίλυση προβλημάτων. Σε ένα σημαντικό βαθμό αυτές οι αρχές θα έπρεπε να είναι ανεξάρτητες από τη γλώσσα προγραμματισμού που χρησιμοποιείται. Κάτι τέτοιο όμως είναι πρακτικά ανέφικτο. Έτσι κάθε γλώσσα περιορίζει σε κάποιο βαθμό την εκφραστική δύναμη των προγραμματιστών αφού εισάγει κάποιους περιορισμούς. Ένα χαρακτηριστικό παράδειγμα είναι η γλώσσα Pascal, ο πιο δημοφιλής εκπρόσωπος του δομημένου προγραμματισμού. Η χρήση της Pascal για δεκαετίες στην εκπαίδευση επέβαλλε στην εκπαιδευτική κοινότητα τις αρχές του δομημένου προγραμματισμού. Πιο συγκεκριμένα επέβαλλε τον περιορισμό ότι κάθε επανάληψη έχει μόνο ένα σημείο εισόδου και το πολύ ένα σημείο εξόδου. Η χρήση εντολών εξόδου όπως η goto από το μέσο του βρόχου ήταν αυστηρά απαγορευμένη ειδικά την δεκαετία του 1980. Για αυτό φρόντισε ο Edsger Dijkstra με το μνημειώδες άρθρο του *Go to statement*

ISBN: 978-960-578-058-6

considered harmful (Dijkstra, 1968). Αρκετοί επιστήμονες αλλά όχι όλοι (Knuth, 1974) συστρατεύθηκαν με τον Dijkstra με αποτέλεσμα τον εξοβελισμό της goto από την ακαδημαϊκή κοινότητα και τη βιομηχανία λογισμικού.

Τα τελευταία χρόνια όμως με την επικράτηση γλωσσών όπως οι Python, Ruby, Javascript, C++, Java και την ολοκληρωτική εξαφάνιση της Pascal από τα εισαγωγικά μαθήματα προγραμματισμού, το θέμα της χρήσης εντολών διακλάδωσης έχει τεθεί πάλι. Η χρήση της εντολής break αποτέλεσε την αφορμή για να συζητηθεί πάλι το θέμα της παράκαμψης κάποιων αρχών του δομημένου προγραμματισμού όχι μόνο για λόγους απόδοσης αλλά κυρίως για διδακτικούς λόγους (Doss, 2014; Smith, Zemljic, & Petersen, 2015; Sorva & Vihavainen, 2016).

Όλες οι έρευνες που έχουν γίνει δείχνουν ότι η δυνατότητα εξόδου από οποιοδήποτε σημείο του βρόχου διευκολύνει (Barnes & Shinners–Kennedy, 2011; Shapiro, 1980; Soloway, Bonar & Ehrlich, 1983) τους αρχάριους προγραμματιστές αφού έτσι αποφεύγουν τη σύνταξη πολύπλοκων συνθηκών τερματισμού μιας επαναληπτικής δομής.

Ένα άλλο επιχείρημα που αναπτύσσεται και σε αυτή την εργασία είναι ότι οι εντολές διακλάδωσης είναι πολύ πιο κοντά στον ανθρώπινο τρόπο σκέψης (Knuth, 1974; Roberts, 1993;1995) από ότι οι εντολές επανάληψης while/return του δομημένου προγραμματισμού. Μια υψηλού επιπέδου ψευδογλώσσα θα έπρεπε να υποστηρίζει αυτές τις εντολές. Αυτό όμως σε πολλές περιπτώσεις δε συμβαίνει ακόμα, γιατί ένα μεγάλο μέρος της εκπαιδευτικής κοινότητας ήταν για δεκαετίες δέσμιο των στερεοτύπων του δομημένου προγραμματισμού.

Σε αυτή την εργασία προσπαθούμε να υπερβούμε τα στερεότυπα αυτά τα οποία είναι βαθιά ριζωμένα σε ένα μεγάλο μέρος της εκπαιδευτικής κοινότητας της πληροφορικής. Για το σκοπό αυτό παρουσιάζουμε τα προκαταρκτικά αποτελέσματα μιας έρευνας που πραγματοποιήθηκε σε πρωτοετείς φοιτητές του τμήματος Μηχανικών Βιοϊατρικής του Πανεπιστημίου Δυτικής Αττικής στο πλαίσιο του μαθήματος ‘Τεχνικές Προγραμματισμού’ το οποίο αποτελεί μια εισαγωγή στον προγραμματισμό με τη γλώσσα Python. Η ίδια έρευνα έγινε και σε 15 μαθητές γυμνασίου. Οι φοιτητές χωρίστηκαν σε δυο ομάδες, την ομάδα ελέγχου η οποία διδάχθηκε με την δομημένη προσέγγιση και την πειραματική ομάδα η οποία είχε τη δυνατότητα να χρησιμοποιήσει την εντολή break για την ανάπτυξη προγραμμάτων. Ερευνήσαμε αν οι φοιτητές αποφεύγουν τη χρήση αυτών των εντολών ή τις προτιμούν έναντι δομημένων αλγορίθμων. Η ανάλυση των αποτελεσμάτων είναι κυρίως ποιοτική και λιγότερο ποσοτική. Τα συμπεράσματα της έρευνας συμφωνούν με τα αποτελέσματα όλων των διεθνών ερευνών και δείχνουν την χρησιμότητα των εντολών διακλάδωσης σε ένα εισαγωγικό μάθημα προγραμματισμού. Ωστόσο εντοπίστηκε μια σημαντική παρανόηση που έχουν οι φοιτητές όταν χρησιμοποιούν την εντολή break εντός ένθετων βρόχων, η οποία δεν έχει αναφερθεί μέχρι στιγμής

ISBN: 978-960-578-058-6

στην βιβλιογραφία. Η παρανόηση αυτή διερευνήθηκε περαιτέρω μέσω συνεντεύξεων με τους φοιτητές και τα αποτελέσματα παρουσιάζουν ενδιαφέρον.

2. Ιστορική Εξέλιξη

Η κατάχρηση της εντολής goto, είχε σαν αποτέλεσμα την δημιουργία μη αναγνώσιμων προγραμμάτων τα οποία ήταν πρακτικά αδύνατο να συντηρηθούν ακόμα και από εκείνους που τα έγραψαν. Η πρώτη αντίδραση της επιστημονικής κοινότητας ήταν η θεωρητική δουλειά των Bohm και Jacopini (1966), που αποτελούσε βελτίωση μιας δημοσίευσης που είχαν ήδη κάνει το 1964 σε ένα συνέδριο στο Ισραήλ. Ωστόσο η δουλειά αυτή έτυχε αναγνώρισης από όλο τον κόσμο κυρίως μετά τη φημισμένη πρόταση Go to statement considered harmful του E.W. Dijkstra (Dijkstra, 1968). Ο Dijkstra δεν είχε κάποια θεωρητική απόδειξη όπως οι Bohm και Jacopini, είχε όμως πειστικά επιχειρήματα ότι η αλόγιστη χρήση της εντολής goto μπορεί να δημιουργήσει μεγάλα προβλήματα τα οποία αυξάνονται όσο αυξάνεται το μέγεθος του προγράμματος. Τη δεκαετία του 70' με την εξάπλωση γλωσσών που προήγαγαν τον δομημένο προγραμματισμό όπως η ALGOL και αργότερα η PASCAL, άρχισε να περιορίζεται σημαντικά η χρήση της εντολής goto.

Η συζήτηση για τη χρήση ή μη εντολών διακλάδωσης ξανάνοιξε όταν η C διαδέχθηκε την Pascal στα περισσότερα εισαγωγικά μαθήματα προγραμματισμού των πανεπιστημιακών τμημάτων. Το 1992 ο Eric Roberts, καθηγητής στο πανεπιστήμιο του Stanford, σχεδίασε ένα νέο εισαγωγικό μάθημα προγραμματισμού με τη γλώσσα C (Roberts, 1993), κάτι που φάνταζε αρκετά προχωρημένο για εκείνη την εποχή. Στην αποτίμηση αυτής της προσπάθειας (Roberts, 1993) αναφέρει ότι οι διδάσκοντες δεν παρατήρησαν κακή ή υπερβολική χρήση των εντολών break και return όπως ίσως αυτή που είχε στο μυαλό του ο Dijkstra για τη goto. Στη συνέχεια ο Roberts εξέδωσε και ένα βιβλίο βασισμένο στην διδακτική του προσέγγιση(Roberts, 1995b) με τίτλο 'The Art and Science of C: A Library-Based Approach', το οποίο έμελλε να εξελιχθεί σε ένα από πιο δημοφιλέστερα βιβλία προγραμματισμού στον κόσμο.

Ο Roberts μπορεί να δικαιώθηκε μετά από χρόνια αλλά εκείνη την εποχή ο δομημένος προγραμματισμός και πιο συγκεκριμένα η Pascal είχαν εξελιχθεί σε στερεότυπα που δεν τολμούσε να αμφισβητήσει κανείς. Έτσι δικαιολογούνται αρκετές αρνητικές κριτικές που έλαβαν οι εργασίες και το βιβλίο του Roberts από επιστήμονες της εποχής. Υπήρξε μάλιστα και δημοσιευμένη εργασία με τίτλο 'C in the First Course Considered Harmful' (Johnson, 1995) με πολύ αρνητικά σχόλια για την διδακτική προσέγγιση του Roberts, όχι μόνο αναφορικά με τη break αλλά γενικότερα για τη γλώσσα C.

Σύμφωνα με τον Roberts(1995) η μετάβαση από την Pascal στη C ήταν επιβεβλημένη όχι μόνο εξαιτίας της ευρείας διάδοσης της C αλλά και λόγω των περιορισμών που επέβαλε το δομημένο μοντέλο της Pascal. Ο Roberts συμπέρανε, από την πολυετή διδακτική εμπειρία του, ότι πολλοί φοιτητές δυσκολεύονταν πολύ στην ανάπτυξη

ISBN: 978-960-578-058-6

προγραμμάτων όπου ο τερματισμός ενός βρόχου ή ο τερματισμός μιας συνάρτησης οφειλόταν σε περισσότερα από ένα γεγονότα. Η διατύπωση της σύνθετης συνθήκης της δομής επανάληψης σε συνδυασμό με την επιλογή του σημείου εξόδου από τον βρόχο (αρχή ή τέλος), δημιουργούσε μεγάλες δυσκολίες στους φοιτητές. Η λύση σε αυτό το προγραμματιστικό και όχι αλγοριθμικό πρόβλημα είναι κατά τη γνώμη του Roberts, η χρήση δομών επανάληψης που επιτρέπουν την έξοδο από οποιοδήποτε σημείο του βρόχου. Στο ίδιο συμπέρασμα καταλήγει και ο Ben-Ari(1996) σύμφωνα με τον οποίο η λελογισμένη χρήση των εντολών break και return μπορεί να διευκολύνει τον έλεγχο ορθότητας ενός προγράμματος και την κατανόηση της λειτουργίας του.

3. Επισκόπηση της ερευνητικής περιοχής

Μέχρι και σήμερα έχουν γραφτεί πολλά άρθρα και έχουν διατυπωθεί πολλές απόψεις σχετικά με τη διδασκαλία ή μη της εντολής διακλάδωση break. Αυτό όμως που έχει σημασία είναι τα επιστημονικά δεδομένα και οι έρευνες που έχουν γίνει. Η πρώτη μεγάλη έρευνα έγινε στο πανεπιστήμιο του Yale στις αρχές του 1980 (Soloway, Bonar, & Ehrlich, 1983). Οι ερευνητές χώρισαν τους φοιτητές σε δυο ομάδες την ομάδα ελέγχου που χρησιμοποιούσε την Standard Pascal και την πειραματική ομάδα που χρησιμοποιούσε την Pascal L η οποία περιλάμβανε μια δομή επανάληψης (loop…again) με εντολή διακλάδωσης (leave) για πρόωρη έξοδο από τον βρόχο. Παρόλο που οι φοιτητές της πειραματικής ομάδας δεν γνώριζαν την Pascal L πριν την έρευνα αφού τους έγινε μια ολιγόλεπτη παρουσίαση των δυνατοτήτων της λίγο πριν τη χρησιμοποιήσουν, τα πήγαν πολύ καλύτερα από τους φοιτητές της ομάδας ελέγχου. Μάλιστα τα αποτελέσματα ήταν τόσο συντριπτικά υπέρ της πειραματικής ομάδας που εξέπληξαν ακόμα και τους ερευνητές. Με αυτό τον τρόπο η ομάδα του Soloway κατέληξε σε δυο σημαντικά συμπεράσματα. Πρώτον ότι οι φοιτητές προτιμούν τη break έναντι μιας δομημένης while και δεύτερον ότι η χρήση της break οδηγεί σε προγράμματα με λιγότερα λάθη από ότι αν κάποιος ακολουθούσε τη δομημένη προσέγγιση.

Τα ίδια και καλύτερα αποτελέσματα είχε μια άλλη έρευνα που έγινε την ίδια εποχή από τον Henry Shapiro(1980) η οποία δυστυχώς δεν είχε την προσοχή που της άξιζε από την ερευνητική κοινότητα παρά μόνο πολύ αργότερα. Στην έρευνα αυτή όλοι οι φοιτητές της πειραματικής ομάδας έλυσαν σωστά το πρόβλημα που τους δόθηκε ενώ από τους φοιτητές της ομάδας ελέγχου το έλυσαν σωστά μόνο το 17%. Σύμφωνα με τον Shapiro οι φοιτητές της ομάδας ελέγχου δυσκολεύτηκαν τόσο πολύ γιατί εργάστηκαν στο περιοριστικό πλαίσιο της Pascal το οποίο δεν επιτρέπει εντολές διακλάδωσης για πρόωρη έξοδο από ένα βρόχο.

Σήμερα πολλοί ερευνητές (Barnes, & Shinners–Kennedy, 2011; Doss, 2014; Sorva, & Vihavainen, 2016; Smith, Zemljic, & Petersen, 2015) προτείνουν τη χρήση των break και return σε εισαγωγικά μαθήματα προγραμματισμού. Τα δυο βασικά

ISBN: 978-960-578-058-6

επιχειρήματα που χρησιμοποιούνται και τώρα είναι η χρησιμότητά τους στην υλοποίηση δύσκολων βρόχων, κάτι που διευκολύνει πολύ τους αρχάριους προγραμματιστές και η ευκολία με την οποία μπορεί να γίνει η απόδειξη της ορθότητας του αλγορίθμου. Το ερώτημα εδώ είναι ότι αφού όλα τα ερευνητικά δεδομένα είναι συντριπτικά υπέρ της χρήσης των εντολών διακλάδωσης break και return, γιατί ακόμα και σήμερα αρκετοί καθηγητές πληροφορικής θεωρούν κακή πρακτική τη χρήση τους;

4. Ανάλυση του Προβλήματος

Ο Knuth(1974) παρατηρεί ότι ναι μεν υπάρχουν δυο τύποι δομών επανάληψης (while/repeat) που έχουν τη συνθήκη τερματισμού στην αρχή ή στο τέλος της επανάληψης, αλλά θα μπορούσαμε να ορίσουμε έναν ακόμα γενικότερο τύπο που περιλαμβάνει όλες τις περιπτώσεις.

```
A: Επανάλαβε                          Επανάλαβε
       Εντολές_πριν                       Εντολές_πριν
       Αν Συνθήκη Τότε Πήγαινε στο B      Αν Συνθήκη Τότε Έξοδος
       Εντολές_μετά                       Εντολές_μετά
   Πήγαινε στο A                      Τέλος_Επανάληψης
B:
```

Αριστερά δίνουμε την πρωτότυπη έκδοση όπως αναφέρεται στο άρθρο του Knuth το 1974 και δίπλα τη δική μας έκδοση που ταιριάζει περισσότερο στην ψευδογλώσσα που διδάσκεται στην ελληνική εκπαίδευση. Σε περίπτωση που δεν υπάρχουν εντολές πριν τη συνθήκη έχουμε τη δομή επανάληψης Όσο (while) ενώ όταν δεν υπάρχουν εντολές μετά τη συνθήκη έχουμε τη δομή επανάληψης Μέχρις_ότου (repeat..until). Τα προβλήματα δημιουργούνται όταν έχουμε εντολές πριν και μετά τον έλεγχο της συνθήκης.

Το πρόβλημα αυτό είναι γνωστό ως "*loop-and-a-half-problem*" και οφείλει το όνομά του στον Edsger Dijkstra. Αυτό είναι το σημείο στο οποίο δυσκολεύονται πολύ οι αρχάριοι προγραμματιστές γιατί χρειάζονται να κάνουν διάφορες αλχημείες, όπως για παράδειγμα να αντιγράψουν τις εντολές πριν τη συνθήκη έξω από την επανάληψη. Αυτό γίνεται όταν η συνθήκη εξαρτάται από τις εντολές που εκτελούνται πριν:

```
Εντολές_πριν
Όσο όχι Συνθήκη Επανάλαβε
    Εντολές_μετά
    Εντολές_πριν
Τέλος_Επανάληψης
```

ISBN: 978-960-578-058-6

Ένα σημαντικό πρόβλημα του παραπάνω τμήματος κώδικα είναι ότι περιέχει δυο φορές τις ίδιες εντολές. Επίσης είναι φανερό ότι απέχει πολύ από τον ανθρώπινο τρόπο σκέψης και την ιδανική ψευδογλώσσα που θα θέλαμε διότι περιορίζεται από τη γλώσσα προγραμματισμού που έχουμε επιλέξει (PASCAL-like), ενώ η προηγούμενη έκδοση είναι ακριβώς το αντίθετο.

Η λύση που προτείνουν αρκετοί ερευνητές της πληροφορικής εκπαίδευσης (Ben-Ari, 1996; Knuth, 1974; Roberts, 1995) είναι η χρήση μιας μόνο δομής επανάληψης η οποία με χρήση μιας εντολής break θα διακόπτεται σε όποιο σημείο θέλει ο προγραμματιστής. Η δομή αυτή συναντάται στη γλώσσα Ada. Ένα χαρακτηριστικό παράδειγμα που χρησιμοποιούν οι ερευνητές για να τεκμηριώσουν τον ισχυρισμό τους είναι αυτό της γραμμικής/σειριακής αναζήτησης ενός στοιχείου σε έναν πίνακα.

```
Αλγόριθμος Αναζήτηση( Α, τιμή)
    i ← 1
    Βρέθηκε ← Ψευδής
    Όσο i<=N και Βρέθηκε=Ψευδής Επανάλαβε
        Αν A[ i ] = τιμή Τότε
            Βρέθηκε ← Αληθής
        i ← i + 1
    Αναζήτηση ← Βρέθηκε
```

```
Αλγόριθμος Αναζήτηση( Α, τιμή)
    Για i από 1 μέχρι N
        Αν A[ i ] = τιμή Τότε
            Αναζήτηση ← Αληθής

    Αναζήτηση ← Ψευδής
```

Σύμφωνα με τα αποτελέσματα όλων των ερευνών, η χρήση της εντολής διακλάδωσης παρέχει στους αρχάριους προγραμματιστές έναν τρόπο αποφυγής της σύνθετης συνθήκης η οποία τους δυσκολεύει πολύ.

Επίσης όπως δείξαμε και παραπάνω η διατύπωση του αλγορίθμου είναι πιο κοντά σε ψευδογλώσσα η οποία είναι πολύ πιο κοντά στον τρόπο που διατυπώνουν τη σκέψη τους οι άνθρωποι. Για παράδειγμα όταν θέλουμε να δώσουμε οδηγίες θα πούμε : "Προχώρα ευθεία, όταν δεις ένα σχολείο σταμάτα" και όχι "Όσο δεν βλέπεις ένα σχολείο, προχώρα ευθεία".

5. Μεθοδολογία της έρευνας

Η έρευνα που παρουσιάζουμε σε αυτή την εργασία έγινε σε 40 πρωτοετείς φοιτητές του τμήματος Μηχανικών Βιοϊατρικής του Πανεπιστημίου Δυτικής Αττικής και σε 15 μαθητές του Ζαννείου Πειραματικού Γυμνασίου που συμμετείχαν στον όμιλο Αλγοριθμικής. Όσον αφορά τους φοιτητές, η έρευνα έλαβε χώρα στο πλαίσιο του μαθήματος Τεχνικές Προγραμματισμού του εαρινού εξαμήνου 2018-2019 στο οποίο οι φοιτητές εισάγονται στην αλγοριθμική σκέψη με τη γλώσσα προγραμματισμού Python. Όλοι οι φοιτητές προέρχονται από την θετική κατεύθυνση και έχουν διδαχθεί

ISBN: 978-960-578-058-6

το μάθημα Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον στην Γ' Λυκείου, το οποίο όμως δεν ήταν πανελλαδικά εξεταζόμενο στην δική τους κατεύθυνση. Οι μαθητές από την άλλη πλευρά είχαν μια πρώτη επαφή με τη γλώσσα προγραμματισμού Python στο πλαίσιο του ομίλου Αλγοριθμικής ενός πειραματικού γυμνασίου ο οποίος διεξάγεται μετά το πέρας των μαθημάτων και στον οποίο οι μαθητές επιλύουν αλγοριθμικά προβλήματα στις γλώσσες C++ και Python.

Ο βασικός στόχος της έρευνας ήταν η διερεύνηση της στάσης των φοιτητών και των μαθητών αναφορικά με τις εντολές διακλάδωσης break και return και τις δυσκολίες που συναντούν στη χρήση τους. Ένα ερώτημα που διερευνήθηκε ήταν αν η χρήση της εντολής break οδηγεί σε πιο ορθά προγράμματα σε σχέση με την προσέγγιση του δομημένου προγραμματισμού;

Οι φοιτητές ήταν χωρισμένοι σε δυο ομάδες από την αρχή της χρονιάς. Στην ομάδα ελέγχου δεν παρουσιάστηκε κανένα παράδειγμα χρήσης της εντολής break ούτε εξόδου μέσα από βρόχο με την εντολή return. Αντίθετα η πειραματική ομάδα είχε τη δυνατότητα να χρησιμοποιεί την εντολή break όποτε ήθελε αλλά μπορούσαν να ακολουθούν και τη δομημένη προσέγγιση. Σε κάθε ομάδα δόθηκαν 5 προβλήματα σε δυο ώρες στο εργαστήριο και τους ζητήθηκε να σχεδιάσουν μια λύση χωρίς να εκτελούνται περιττοί υπολογισμοί.

6. Αποτελέσματα

Λόγω του περιορισμένου χώρου που έχουμε στο άρθρο θα παρουσιάσουμε κάποια προκαταρκτικά αποτελέσματα της έρευνας που θεωρούμε ότι έχουν ενδιαφέρον.

Στο πρώτο πρόβλημα ζητήθηκε η υλοποίηση της σειριακής αναζήτησης ενός στοιχείου σε μια λίστα με τη μορφή συνάρτησης. Όλοι οι φοιτητές της πειραματικής ομάδας έδωσαν μια σωστή συνάρτηση σε Python με χρήση της return, ενώ στην ομάδα ελέγχου οι μισοί φοιτητές έκαναν κάποια λάθη στη συνθήκη τερματισμού. Από αυτούς πέντε φοιτητές έδωσαν την παρακάτω λύση η οποία προϋποθέτει ότι η γλώσσα υποστηρίζει *πρόωρη αποτίμηση* (*short-circuit evaluation)*, που δεν ισχύει.

Ομάδα Ελέγχου

```
def search(A, key):
    i = 0
    while i<len(A) and A[i]!=key :
        i = i + 1

    return A[i]==key
```

Πειραματική Ομάδα

```
def search(A, key):
    for item in A:
        if item==key:
            return True
    return False
```

Οι φοιτητές της ομάδας ελέγχου οδηγήθηκαν στον παραπάνω αλγόριθμο στην προσπάθειά τους να αποφύγουν τη χρήση λογικής μεταβλητής, κάτι που επιβεβαιώσαμε αργότερα στις συνεντεύξεις που είχαμε μαζί τους. Εδώ αξίζει να

ISBN: 978-960-578-058-6

σημειωθεί ότι ο αλγόριθμος που έδωσαν οι φοιτητές της πειραματικής ομάδας είναι πολύ πιο κοντά στον ανθρώπινο τρόπο σκέψης και θα μπορούσε να χαρακτηριστεί ως ψευδοκώδικας υψηλού επίπεδου αφαίρεσης.

Στο δεύτερο πρόβλημα τους ζητήθηκε να αναπτύξουν έναν αλγόριθμο ο οποίος θα διαβάζει το πολύ 100 θετικούς αριθμούς και θα υπολογίζει τον μέσο όρο τους. Ο αλγόριθμος θα τερματίζει επίσης αν δοθεί μη θετικός αριθμός. Πρόκειται για το γνωστό πρόβλημα "*loop-and-a-half-problem*" που αναλύθηκε στην προηγούμενη παράγραφο. Παρακάτω δίνονται οι αποδεκτές λύσεις για κάθε ομάδα.

Ομάδα Ελέγχου

```
number = int(input())
S=i=0
while i<100 and number>0:
    i+=1
    S+=number
    number = int(input())
avg = S / i
print(avg)
```

Πειραματική Ομάδα

```
S=i=0
while i<100:
    number = int(input())
    if number<=0:
        break
    i+=1
    S+=number
avg = S / i
print(avg)
```

Η διάφορα έγκειται στο γεγονός ότι οι 18 στους 20 φοιτητές της πειραματικής ομάδας έδωσαν σωστή λύση ενώ από την ομάδα ελέγχου μόνο 5 στους 20 έδωσαν αλγόριθμο χωρίς σφάλματα με βάση τις αρχές του δομημένου προγραμματισμού. Έχει ενδιαφέρον να σταθούμε σε μια ακόμα δραστηριότητα την αναζήτηση σε πίνακα δυο διαστάσεων η οποία ζητήθηκε να δοθεί ως πρόγραμμα και όχι ως συνάρτηση έτσι ώστε η ομάδα ελέγχου να υποχρεωθεί να χρησιμοποιήσει την break και όχι την return. Ενώ όλοι οι φοιτητές έδειξαν ότι καταλαβαίνουν τη λειτουργία της return, δίσταζαν να χρησιμοποιήσουν τη break σε ένθετους βρόχους. Στην συζήτηση που ακολούθησε φάνηκε ότι οι φοιτητές δεν ήταν σίγουροι για το σημείο μετάβασης του ελέγχου μετά την break. Δηλαδή αν θα συνέχιζε μέσα στην εξωτερική επανάληψη ή αν θα βρισκόταν έξω και από τις δυο επαναλήψεις. Αυτή είναι μια δυσκολία που δεν περιμέναμε να συναντήσουμε και ούτε έχει καταγραφεί στην βιβλιογραφία πιθανόν επειδή όλες οι έρευνες που έχουν γίνει δεν είχαν προβλήματα με ένθετους βρόχους. Την ίδια ακριβώς δυσκολία συνάντησαν και οι μαθητές του ομίλου αλγοριθμικής.

7. Συμπεράσματα

Ο Eric Roberts (1995) ισχυρίζεται ότι δεν χρειάζεται να περιορίζουμε την έξοδο από την αρχή ή το τέλος ενός βρόχου. Αν κριθεί χρήσιμο τότε η έξοδος θα μπορούσε να γίνει και από άλλο σημείο του βρόχου, π.χ. από το μέσο του. Το ίδιο ισχύει και για την επιστροφή από μια συνάρτηση στο κύριο πρόγραμμα. Δεν χρειάζεται η εντολή return να είναι πάντα η τελευταία εντολή της συνάρτησης.

ISBN: 978-960-578-058-6

Η ανάλυση του Roberts βασίζεται στο γεγονός ότι η υπερβολική προσήλωση στον δομημένο προγραμματισμό μπορεί να δημιουργήσει αρκετά πολύπλοκα και δυσανάγνωστα προγράμματα κάτι που θέλουμε να αποφύγουμε ειδικά στην εκπαίδευση. Θα πρέπει λοιπόν να αναθεωρήσουμε ριζικά το υπάρχον παιδαγωγικό πλαίσιο της διδασκαλίας του προγραμματισμού όσον αφορά την εμμονή μας σε στερεότυπα του δομημένου προγραμματισμού.

Όταν οι φοιτητές αναπτύσσουν λύσεις σε γλώσσες που περιορίζονται από τις αρχές του δομημένου προγραμματισμού, παρουσιάζουν δυσκολίες στην ανάπτυξη τους. Η έρευνα που έγινε σε αυτή την εργασία σε πρωτοετείς φοιτητές του τμήματος Βιοϊατρικής του Πανεπιστημίου Δυτικής Αττικής έδειξε ότι οι φοιτητές που είχαν τη δυνατότητα χρήσης των εντολών break και return είχαν πολύ καλύτερη απόδοση από τους φοιτητές της ομάδας ελέγχου και συνάντησαν πολύ λιγότερες δυσκολίες κατά την ανάπτυξη των αλγορίθμων τους σε Python. Ωστόσο υπήρχε μια δυσκολία στην χρήση της break όταν αυτή βρίσκεται μέσα σε ένθετους βρόχους. Σε αυτή την περίπτωση αρκετοί φοιτητές δήλωσαν ότι δεν είναι σίγουροι σε ποια εντολή θα τους στείλει η break, δηλαδή αν θα τερματίσουν και οι δυο επαναλήψεις ή μόνο αυτή στην οποία ανήκει η break. Το ίδιο πρόβλημα αντιμετώπισαν και οι μαθητές Γυμνασίου.

Στα ίδια συμπεράσματα για τη διδακτική χρησιμότητα των εντολών break και return έχουν καταλήξει και άλλες έρευνες κάποιες από τις όποιες έγιναν τη δεκαετία του 80. Δεν έλαβαν όμως την προσοχή της επιστημονικής κοινότητα λόγω της εμμονής πολλών ερευνητών στα στερεότυπα του δομημένου προγραμματισμού και της Pascal που δεν τόλμησε να αμφισβητήσει κανείς. Ήρθε όμως η ώρα να τεθούν και αυτά στο χρονοντούλαπο της ιστορίας γιατί ο λόγος ύπαρξής τους έχει πλέον εκλείψει.

Αναφορές

Barnes, D. J. & Shinners – Kennedy. D. (2011). A study of loop style and abstraction in pedagogic practice. In Proceedings of the 13th Australasian Conference on Computing Education (ACE ‘11), Volume 114 of CRPITS, 29-36.

Ben-Ari, M. (1996). Structure Exits, Not Loops, *SIGCSE Bulletin, 28*(3).

Bohm, C. & Jacopini, G. (1966), Flow diagrams, Turing machines and languages with only two formation rules, *Communications of the ACM, 9*(5):366-371.

Dijkstra, E.W. (1968). Go to statement considered harmful. *Communications of the ACM, 11*:147-148.

ISBN: 978-960-578-058-6

Doss, R. (2014). Using Goto Statements. *ACM SIGSOFT Software Engineering Notes, 39*(5), 1-3.

Knuth, E. D. (1974). Structured programming with go to statements. *Computing Surveys, 6*(4): 261-301.

Roberts, S. E. (1993). Using C in CS1: Evaluating the Stanford experience. SIGCSE Bulletin.

Roberts, S. E. (1995). Loop exits and structure programming: Reopening the debate. 26th SIGCSE Technical Symposium, 268-272.

Roberts, S. E. (1995b). *The Art and Science of C: A Library-Based Approach*. Reading, MA: Addison-Wesley.

Shapiro, H. (1980). The results of an informal study to evaluate the effectiveness of teaching structured programming, *ACM SIGCSE Bulletin, 12*(4), 50-56.

Smith, S. D., Zemljic, N., & Petersen, A. (2015) Modern goto: Novice programmer usage of non-standard control flow. In: Proceedings of the 15th Koli Calling Conference on Computing Education Research, Koli Calling.

Soloway E., Bonar J., & Ehrlich K. (1983). Cognitive Strategies and looping constructs: an empirical study. *Communications of the ACM, 26*(11).

Sorva, J. & Vihavainen, A. (2016). Break Statement Considered. *ACM Inroads, 7*(3), 36-41.

Abstract

The goto jump command has been sidelined for many years at both educational and professional levels, while other branching commands such as break and return are now widely used in application development and education. However, the use of these commands is not yet accepted by the educational community. The key question is whether these commands have anything to offer in the teaching of programming and algorithms or they are the source of difficulties for students in introductory programming courses. In this context, we conducted a field survey of 40 first-year students and 14 high school students in order to explore the difficulties that students face with these commands. We investigated whether students avoid using these commands or prefer structured algorithms. Our conclusions are consistent with the results of all the corresponding surveys and show the usefulness of branching commands in an introductory programming course. However, a significant misunderstanding has been identified by students when using the break command in nested loops, which has not been reported in the literature so far.

Keywords: branching statements, break, return, python, structured programming.

ISBN: 978-960-578-058-6