Η εύρεση μέγιστου στοιχείου ως ολοκληρωμένο παράδειγμα εφαρμογής υπολογιστικής σκέψης

Παναγιώτης Γροντάς

Καλλιτεχνικό Γυμνάσιο Γέρακα με Λυκειακές Τάξεις, pgrontas@gmail.com

Περίληψη

Η εργασία αυτή παρουσιάζει μια πρόταση διδασκαλίας για την εύρεση της ελάχιστης ή μέγιστης τιμής από ένα σύνολο στοιχείων υπό το πρίσμα αρχών της υπολογιστικής σκέψης. Το συγκεκριμένο πρόβλημα είναι αρκετά απλό και οικείο στους μαθητές με αποτέλεσμα να μπορεί να αναλυθεί χωρίς να χρειάζονται περισσότερες γνώσεις και αναπαραστάσεις που η αφομοίωση τους να απομακρύνει από τον στόχο. Επιπλέον είναι ένα παράδειγμα στο οποίο μπορεί να γίνουν κατανοητές με βιωματικό τρόπο αρχές της υπολογιστικής σκέψης σε όλα τα στάδια της επίλυσής του: από την αρχική ανάλυση μέχρι την κωδικοποίηση.

Λέξεις κλειδιά: Υπολογισμός Ελάχιστης\Μέγιστης τιμής, υπολογιστική σκέψη, ανάλυση προβλήματος, αναδρομική επίλυση, αναγνώριση προτύπων

1. Εισαγωγή

Η υπολογιστική σκέψη είναι ένα σύνολο δεξιοτήτων και στάσεων χρήσιμο σε κάθε μαθητή (Wing, 2006). Δραστηριότητες υπολογιστικής σκέψης έχουν προταθεί για διάφορες ηλικίες (Ζερβουδάκη & Παπαδάκης, 2018) και μαθήματα (Κοταρίνου, Κουλέτση, Πλιάκου, Συριόπουλος, & Χούπη, 2018). Όμως, το διδακτικό αντικείμενο της πληροφορικής είναι ο κατεξοχήν αντιπρόσωπός της στα διάφορα εκπαιδευτικά συστήματα, αφού η υπολογιστική σκέψη αποτελεί το κυριότερο εργαλείο της, ως ένα χαρακτηριστικό που αποκτούν όλοι οι επιστήμονες της πληροφορικής, είτε το έχουν κατακτήσει συνειδητά είτε ασυνείδητα. Με αυτό το πρίσμα, έννοιες της υπολογιστικής σκέψης υπάρχουν σε όλα τα προγράμματα σπουδών πληροφορικής της δευτεροβάθμιας εκπαίδευσης, έστω και αν δεν γίνεται ρητή αναφορά σε αυτήν.

Από την άλλη, η εύρεση του ελάχιστου ή του μέγιστου[1] από ένα σύνολο στοιχείων είναι ένα βασικό αλγοριθμικό πρόβλημα. Εμπειρικά έχει διαπιστωθεί πως όταν τα στοιχεία είναι πολλά έχει επικρατήσει ένας επαναληπτικός αλγόριθμος για την επίλυση του προβλήματος, τον οποίο οι περισσότεροι μαθητές κατανοούν και χρησιμοποιούν χωρίς πρόβλημα. Το αξιοπερίεργο όμως είναι ότι όταν τα στοιχεία προς εξέταση είναι λιγότερα (3 ή 4), οι μαθητές δεν χρησιμοποιούν μια εξειδίκευση του, αλλά επινοούν

[1] Η προσέγγιση του άρθρου μπορεί να εφαρμοστεί και στις δύο περιπτώσεις. Για λόγους απλότητας θα αναφερόμαστε από εδώ και στο εξής στο μέγιστο στοιχείο.

έναν διαφορετικό ο οποίος έχει αρκετές παγίδες, όπως θα δούμε, και δεν γενικεύεται εύκολα.

Στην εργασία αυτή επεμβαίνουμε και στα δύο παραπάνω ζητήματα: Δίνουμε ένα ολοκληρωμένο παράδειγμα εφαρμογής της υπολογιστικής σκέψης χρησιμοποιώντας ως πρόβλημα αυτό της εύρεσης του μέγιστου από κάποια στοιχεία. Η προσέγγιση μας έχει δύο πλεονεκτήματα: Πρώτα από όλα κάνει πιο απτή την διδασκαλία της υπολογιστικής σκέψης. Αυτό συμβαίνει δεν αναφέρεται στις αρχές της θεωρητικά, αλλά τις εφαρμόζει σε ένα απλό παράδειγμα το οποίο είναι προσιτό στους μαθητές. Επιπλέον όμως οδηγεί σε έναν ολοκληρωμένο (και ορθό σε όλες τις περιπτώσεις τρόπο) για την προσέγγιση του προβλήματος της εύρεσης του μέγιστου από ένα σύνολο στοιχείων.

2. Υπολογιστική σκέψη

Αρχικά θα παρουσιάσουμε κάποιες βασικές αρχές της υπολογιστικής σκέψης. Φυσικά δεν στοχεύουμε σε μια εξαντλητική περιγραφή, αλλά στο να θέσουμε κάποιες βάσεις που θα αξιοποιήσουμε στις επόμενες ενότητες.

2.1 Βασικά στοιχεία

Ο όρος «υπολογιστική σκέψη» πρωτοχρησιμοποιήθηκε στο (Wing, 2006) ως «μία μέθοδος επίλυσης προβλημάτων, σχεδιασμού συστημάτων και κατανόησης της ανθρώπινης συμπεριφοράς που χρησιμοποιεί έννοιες από την επιστήμη των υπολογιστών».

Κάποια από αυτά τα νοητικά εργαλεία είναι (Wing, 2006), (Νείρος & Ζάχαρης, 2018):

- *Διάσπαση και σύνθεση:* Για να αντιμετωπίσουμε ένα σύνθετο πρόβλημα μπορούμε να το χωρίσουμε σε απλούστερα, να επιλύσουμε κάθε ένα από αυτά και στη συνέχεια να συνδυάσουμε τις λύσεις τους. Η ευκολία επίλυσης των υποπροβλημάτων αυτών οφείλεται είτε στο ότι εξετάζουν μία μόνο πτυχή του αρχικού προβλήματος, είτε στο ότι ασχολούνται με μικρότερο πλήθος δεδομένων, λύνοντας στην περίπτωση αυτή το πρόβλημα *αναδρομικά*.
- *Γενίκευση:* Επιτρέπει την μετάβαση από την λύση ενός συγκεκριμένου στιγμιότυπου ενός προβλήματος στην γενική λύση. Για παράδειγμα, μπορούμε να μεταβούμε στον γενικό τύπο επίλυσης πρωτοβάθμιας εξίσωσης, έχοντας πρώτα λύσει μία συγκεκριμένη εξίσωση. Όπως γίνεται σαφές, η γενίκευση οδηγεί στην εξέταση περιπτώσεων που δεν εμπίπτουν στα χαρακτηριστικά του αρχικού στιγμιότυπου.
- *Δημιουργία μοντέλων:* Όταν αντιμετωπίζουμε ή μας δίνεται ένα πρόβλημα προς επίλυση, απομονώνουμε τα ουσιαστικά χαρακτηριστικά του, φτιάχνοντας έτσι μια ιδεατή αναπαράσταση, για την οποία είναι πιο εύκολο να βρούμε την λύση.

- *Αφαίρεση*: Επιτρέπει την χρήση ενός μοντέλου για κάποιο (υπο)πρόβλημα ως μαύρο κουτί, χωρίς δηλαδή να γνωρίζουμε επακριβώς ούτε να μας απασχολούν συγκεκριμένες λεπτομέρειες λειτουργίας. Βέβαια, στο τέλος πρέπει να ελέγξουμε αν η λύση μέσω της αφαίρεσης, ανταποκρίνεται στο πραγματικό στιγμιότυπο που αντιμετωπίσαμε αρχικά, ώστε να μην πέσουμε στην παγίδα των "leaky abstractions" (Spolsky, 2002).
- *Αναγνώριση προτύπων*: Στοχεύει στην ανάπτυξη της ικανότητας εύρεσης μοτίβων σε σύνολα από δεδομένα, δηλαδή σχέσεων και συγκεκριμένων εξαρτήσεων μεταξύ μεμονωμένων στοιχείων. Τα μοτίβα αυτά μπορούν να χρησιμοποιηθούν και για τη δημιουργία μοντέλων αλλά και για τον έλεγχο ότι οι λύσεις που δόθηκαν στα προβλήματα είναι ορθές.

2.2 Ένταξη στη διδασκαλία – παραδείγματα

Τα παραπάνω εργαλεία, αλλά και ο ορισμός στο (Wing, 2006) κάνουν φανερή την χρησιμότητα της υπολογιστικής σκέψης σε όλα τα μαθήματα, αλλά και στην καθημερινή ζωή των μαθητών. Όμως δεν αναιρούν τη σημασία της υπολογιστικής σκέψης στην ίδια τη διδασκαλία της πληροφορικής, όπου οι αρχές της μπορούν να χρησιμοποιηθούν ώστε να οδηγηθούμε σε ορθές και κομψές λύσεις (π.χ. μέσω αναδρομικής σκέψης) ή να ελεγχθούν λύσεις που προκύπτουν ως στιγμιαία έμπνευση ή από λάθος κατανόηση του προβλήματος, χρησιμοποιώντας το εργαλείο της αναγνώρισης προτύπων.

Παρ' όλα αυτά δεν υπάρχει κάποιο τέτοιο ολοκληρωμένο παράδειγμα στα διδακτικά πακέτα που χρησιμοποιούνται στη δευτεροβάθμια εκπαίδευση. Στο Γυμνάσιο (Αράπογλου, Μαβόγλου, Οικονομάκος, & Φύτρος, 2006), δίνεται ιδιαίτερη έμφαση στο εργαλείο της Διάσπασης και Σύνθεσης προβλήματος, όπου το πρόβλημα της οργάνωσης μιας εκπαιδευτικής εκδρομής επιλύεται με ανάλυση σε απλούστερα. Το συγκεκριμένο πρόβλημα, ως ένα πρόβλημα της καθημερινής ζωής των μαθητών είναι εύκολο προς κατανόηση και συμβατό με τις εμπειρίες των μαθητών. Ως τέτοιο όμως δεν μπορεί να χρησιμοποιηθεί αυτούσιο στην συνέχεια του μαθήματος, ώστε οι μαθητές να δουν τα πλεονεκτήματα της προσέγγισης με βάση την υπολογιστική σκέψη κατά την ανάπτυξη ενός αλγόριθμου.

Παρόμοια προσέγγιση ακολουθείται και στο μάθημα Ανάπτυξη Εφαρμογών Σε Προγραμματιστικό Περιβάλλον (Βακάλη, και συν., 1999). Ιδιαίτερη έμφαση δίνεται και εδώ στην δομή ενός προβλήματος. Χρησιμοποιείται και εδώ ένα παράδειγμα από την καθημερινή ζωή – το πρόβλημα των ναρκωτικών, για να τονιστεί το κέρδος από την ανάλυση ενός προβλήματος σε απλούστερα. Το ενδιαφέρον εδώ όμως είναι ότι στο συγκεκριμένο βιβλίο υπάρχει και ένα δεύτερο παράδειγμα που εμπίπτει στον χώρο της πληροφορικής, αυτό της επεξεργασίας των αποτελεσμάτων των πανελλαδικών εξετάσεων. Σε αυτό θα μπορούσαν να εφαρμοστούν τεχνικές της υπολογιστικής σκέψης, ώστε οι μαθητές να δουν πώς σχετίζονται η φάση της ανάλυσης με τον κώδικα

που παράγεται. Κάτι τέτοιο όμως δεν γίνεται, ίσως λόγω του εύρους του προβλήματος αλλά και επειδή κάποιες από τις λειτουργίες που προτείνονται δεν είναι εφικτές στη ΓΛΩΣΣΑ (π.χ. δημιουργία γραφημάτων). Τέλος, τα ίδια ισχύουν και στο μάθημα «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ» (Δουκάκης, Δουληγέρης, Καρβουνίδης, Κοίλιας, & Πέρδος, 2014). Η παρουσίαση κάποιων αρχών της υπολογιστικής σκέψης υπάρχει στα εισαγωγικά κεφάλαια, χωρίς όμως αυτές οι αρχές να εφαρμόζονται με κάποιο ολοκληρωμένο παράδειγμα στη συνέχεια.

Το κενό που περιγράψαμε παραπάνω έρχεται να καλυφθεί από αρκετές προτάσεις που έχουν σκοπό να αναδείξουν στην πράξη τις αρχές της υπολογιστικής σκέψης – ενδεικτικά (Κοτίνη & Τζελέπη, 2016), (Νείρος & Ζάχαρης, 2018) κ.ά.. Αν και αυτές βασίζονται σε εφαρμογές που έχουν στόχο να προσελκύσουν το ενδιαφέρον των μαθητών, πολλές φορές αφορούν τομείς με τους οποίους δεν είναι εξοικειωμένοι οι τελευταίοι με αποτέλεσμα να πρέπει να γίνουν επιπλέον βήματα προτού μπουν στην ουσία της υπολογιστικής σκέψης. Πολλές φορές ακόμα κινούνται στο επίπεδο των γρίφων, χωρίς να υπάρχει καμία καθοδήγηση στο πώς οι μαθητές θα οδηγηθούν οργανωμένα προς τη λύση. Τέλος δεν αφορούν κομμάτια της διδακτέας ύλης, ώστε να μπορούν να χρησιμοποιηθούν σε επόμενες ενότητες\τάξεις.

3. Εύρεση μέγιστου

Έχοντας τώρα το απαραίτητο θεωρητικό υπόβαθρο, θα εφαρμόσουμε τις αρχές της υπολογιστικής σκέψης ώστε να καθοδηγήσουμε τους μαθητές προς την καλύτερη (αντικειμενικά) λύση σε ένα απλό πρόβλημα, το οποίο συναντάται πολύ συχνά.

3.1 Το πρόβλημα και η επιθυμητή λύση

Η βασική μορφή του προβλήματος εύρεσης μέγιστης τιμής είναι ότι δίνεται ένα σύνολο από συγκρίσιμα στοιχεία και ζητείται από τους μαθητές η δημιουργία ενός αλγόριθμου για την εύρεση της μεγαλύτερης από αυτές. Στο πρόβλημα αυτό υπάρχουν αρκετές παραλλαγές ανάλογα με το αν είναι γνωστό εκ των προτέρων το πλήθος των τιμών ή και οι ίδιες οι τιμές ή αν εισάγονται σταδιακά και για την εύρεση του πλήθους τους χρησιμοποιείται κάποια ειδική τιμή (τιμή – φρουρός για να χρησιμοποιήσουμε την ‘φροντιστηριακή’ ορολογία του (Βακάλη, και συν., 1999)).

Για τον καλύτερο συνδυασμό της προσέγγισης μας με τις αρχές της υπολογιστικής σκέψης θα επικεντρωθούμε στην παραλλαγή όπου διαβάζουμε σταδιακά τρεις αριθμούς και θέλουμε να βρούμε τον μεγαλύτερο.

Ο αλγόριθμος που θέλουμε να οδηγήσουμε τους μαθητές είναι ο παρακάτω[2]:

[2] Σε όλη την εργασία χρησιμοποιούμε για απλότητα την ψευδογλώσσα του (Βακάλη, και συν., 1999). Είναι αυτονόητο ότι η προσέγγιση μπορεί να προσαρμοστεί σε οποιοδήποτε περιβάλλον (πχ. Scratch, Microworlds Pro, Pencil Code) επιλέξουν οι διδάσκοντες.

```
1:  Διάβασε α
2:  μέγιστο ← α
3:  Διάβασε β
4:  Αν β > μέγιστο τότε
5:       μέγιστο ← β
6:  Τέλος_Αν
7:  Διάβασε γ
8:  Αν γ > μέγιστο τότε
10:      μέγιστο ← γ
11: Τέλος_Αν
```

Αλγόριθμος 1*. Εύρεση μέγιστου 3 αριθμών (επιθυμητός τρόπος)*

Ο παραπάνω αλγόριθμος έχει κάποια σημαντικά πλεονεκτήματα, όπως τα παρακάτω:

- Δεν έχει πρόβλημα κάποιες από τις τιμές είναι ίσες, καθώς σε αυτή την περίπτωση οι εντολές επιλογής μπορεί να μην εκτελεστούν, αλλά το αποτέλεσμα δεν θα αλλάξει.
- Επεκτείνεται εύκολα για περισσότερες από 3 τιμές, προσθέτοντας απλά μία εντολή εισόδου και μία σύγκριση:

```
Διάβασε δ
Αν δ > μέγιστο τότε
     μέγιστο ← δ
Τέλος_Αν
```

Αλγόριθμος 2. *Επέκταση για τέταρτο αριθμό*

- Είναι εύκολο να γενικευτεί για περισσότερους αριθμούς χρησιμοποιώντας τη δομή επανάληψης:

```
Διάβασε α
μέγιστο ← α
Για ι από 2 μέχρι Ν
     Διάβασε α
     Αν α > μέγιστο τότε
          μέγιστο ← α
     Τέλος_αν
Τέλος_επανάληψης
```

Αλγόριθμος 3. *Επέκταση για πολλά στοιχεία συγκεκριμένου πλήθους*

3.2 Προσεγγίσεις μαθητών

Το πρόβλημα το οποίο αποτέλεσε αφορμή για την παρούσα εργασία είναι η εμπειρική παρατήρηση ότι οι μαθητές όταν τους ζητηθεί να επιλύσουν το πρόβλημα της εύρεσης μέγιστου 3 αριθμών σπάνια δίνουν τις παραπάνω λύσεις. Για παράδειγμα, συναντούμε συχνά την:

```
Διάβασε α,β,γ
Αν α > β και α > γ τότε
     μέγιστο ← α
Αλλιώς_αν β > α και β > γ τότε
     μέγιστο ← β
Αλλιώς
     μέγιστο ← γ
Τέλος_Αν
```

Αλγόριθμος 4. *Τυπικός αλλά λανθασμένος τρόπος*

Επίσης είναι συνηθισμένο το γεγονός πολλοί μαθητές να παρουσιάσουν κάποια παραλλαγή, χρησιμοποιώντας εμφωλευμένη δομή επιλογής. Σε αυτή την περίπτωση δεν καταφέρνουν συνήθως να δώσουν μία ολοκληρωμένη απάντηση. Επίσης έχουμε συναντήσει αρκετές περιπτώσεις όπου χρησιμοποιούνται ανεξάρτητες απλές επιλογές αντί μίας πολλαπλής. Οι λύσεις αυτές δεν είναι ορθές καθώς δεν μπορούν να χειριστούν επιτυχώς την περίπτωση ισότητας κάποιων από τους τρεις αριθμούς: Για παράδειγμα, ενώ στην τριάδα (3,4,5) θα εμφανίσουν σωστά αποτελέσματα, στην τριάδα (4,4,1) θα εμφανίσουν ως αποτέλεσμα το 1 ή δεν θα εμφανίσουν τίποτα. Φυσικά αυτό το λάθος μπορεί να διορθωθεί χρησιμοποιώντας τον τελεστή $\geq$ αντί για το $>$. Ακόμα όμως και σε αυτή την περίπτωση δεν γενικεύονται εύκολα. Για παράδειγμα, ο αλγόριθμος για 4 αριθμούς περιλαμβάνει περισσότερες και πιο συνθέτες συνθήκες, ενώ δεν μπορεί να δοθεί ισοδύναμος αλγόριθμος για μεταβλητό πλήθος αριθμών.

4. Διδακτική προσέγγιση

Η διδακτική παρέμβαση που προτείνεται στη συγκεκριμένη εργασία έχει διάρκεια 3 διδακτικών ωρών. Είναι κατάλληλη για εφαρμογή σε μαθητές τόσο της Γ Γυμνασίου όσο και της Β Λυκείου. Στη δεύτερη περίπτωση, επειδή κάποιες έννοιες θα είναι πιο οικείες στους μαθητές η προσέγγιση μπορεί να συντομευτεί κατά μία ώρα. Υποθέτουμε ότι στα πλαίσια μιας εισαγωγής στο μάθημα, έχουν αναφερθεί κάποιες βασικές αρχές της υπολογιστικής σκέψης, όπως αυτές που περιγράψαμε στην ενότητα 2.1. Επίσης θεωρούμε δεδομένο πώς οι μαθητές γνωρίζουν τη δομή επιλογής και τους λογικούς τελεστές, όχι σε βάθος κατ' ανάγκη. Αν θέλουμε να κάνουμε γενίκευση για περισσότερα στοιχεία θα πρέπει να γνωρίζουν και τη δομή επανάληψης. Σε όλη την παρουσίαση που ακολουθεί υποθέτουμε ότι το μάθημα διεξάγεται στο εργαστήριο πληροφορικής και οι μαθητές παρακολουθούν στον πίνακα χρησιμοποιώντας ταυτόχρονα τους υπολογιστές τους (σε ζευγάρια).

4.1 Εισαγωγή και βάση της αναδρομής

Αρχικά εξηγούμε το πρόβλημα στους μαθητές, χρησιμοποιώντας αναλογίες - παραδείγματα με τα οποία είναι πιθανόν να είναι εξοικειωμένοι (Αδαμόπουλος, 2005). Για παράδειγμα θα μπορούσαν να αναφερθούν περιπτώσεις από αθλητικές

δραστηριότητες ή παιχνίδια στον υπολογιστή, όπου κερδίζει εκείνος που φέρει την μέγιστη τιμή σε κάποιο μέγεθος, όπως μέτρα ή σκορ.

Στη συνέχεια αφήνουμε τους μαθητές για ελεγχόμενο χρονικό διάστημα να προβληματιστούν για το πώς θα έλυναν μόνοι τους το πρόβλημα, ώστε να τους πείσουμε πώς η εύρεση του μέγιστου δεν είναι κάτι προφανές. Επειδή οι μαθητές γνωρίζουν πώς να το λύσουν για τους εαυτούς τους, επισημαίνουμε ότι πρέπει να προσπαθήσουν να εκφράσουν τις εσωτερικές τους σκέψεις με σαφή και καθορισμένο τρόπο, ώστε να προκύψει ο αλγόριθμος. Αφού περάσει ο χρόνος αυτός, τους υπενθυμίζουμε τα εργαλεία της διάσπασης του προβλήματος σε απλούστερα και της αναδρομικής επίλυσης. Οπότε τους αφήνουμε να προβληματιστούμε για την ύπαρξη ενός πιο εύκολου αλλά σχετικού προβλήματος.

Εμπειρικά προκύπτει πως οι μαθητές θα απαντήσουν πως η εύρεση του μέγιστου δύο αριθμών είναι πιο εύκολη. Επανερχόμαστε ρωτώντας αν υπάρχει κάτι πιο εύκολο. Εμπειρικά πάλι, έχουμε διαπιστώσει πως οι μαθητές διστακτικά απαντούν - ρωτώντας στην ουσία – πως ακόμα ευκολότερη είναι η εύρεση του μέγιστου ενός αριθμού. Κάποιοι μπορεί να αναρωτηθούν τι νόημα έχει αυτό. Εδώ απαντάμε με ένα παράδειγμα – σε κάποιον αγώνα που έχουν εγκαταλείψει όλοι εκτός από έναν πρέπει να ανακηρύξουμε νικητή. Αφού πείσουμε όλους τους μαθητές, κατοχυρώνουμε την γνώση γράφοντας τις γραμμές 1, 2 από τον **Αλγόριθμος 1**.

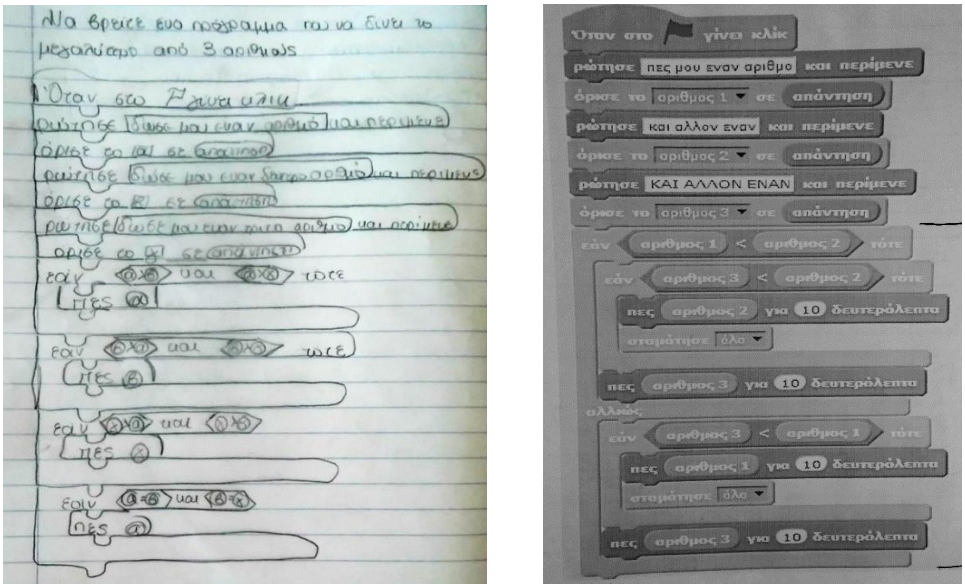
Προσπαθώντας να προετοιμάσουμε τα επόμενα στάδια επιβεβαιώνουμε ότι οι μαθητές γνωρίζουν ότι η λύση του προβλήματος βρίσκεται στη μεταβλητή `μέγιστο`. Στην συνέχεια πρέπει να επιλύσουμε το αμέσως πιο δύσκολο πρόβλημα – δηλαδή την επίλυση του προβλήματος του μέγιστου από δύο αριθμούς. Τονίζουμε ότι πρέπει να εκμεταλλευτούμε ό,τι έχουμε κάνει μέχρι τώρα.

Στο στάδιο αυτό διαπιστώνουμε συνήθως ότι οι μαθητές ξεχνούν τα μέχρι τώρα αποτελέσματα και ξανασκέφτονται το πρόβλημα από την αρχή. Προσπαθούμε να περιορίσουμε κάτι τέτοιο τονίζοντας του ότι πρέπει μέσα στον αλγόριθμο να υπάρχει οπωσδήποτε η χρήση της μεταβλητής `μέγιστο`. Μετά από κάποια συζήτηση προσθέτουμε τις γραμμές 3-6 από τον **Αλγόριθμος 1**.

Σε αυτό το σημείο, σιγουρευόμαστε ότι έχει γίνει κατανοητό ότι και στις τρεις πιθανές περιπτώσεις ($\alpha < \beta$, $\alpha = \beta$, $\alpha > \beta$) οι εντολές που έχουν χρησιμοποιηθεί λύνουν το πρόβλημα σωστά. Αυτό μπορεί να γίνει με συζήτηση στον πίνακα αλλά και εκτέλεση των εντολών στο προγραμματιστικό περιβάλλον που έχει επιλεγεί. Κλείνουμε την πρώτη διδακτική ώρα αναθέτοντας τον αλγόριθμο για την εύρεση του μέγιστου από τρεις αριθμούς ως άσκηση για το σπίτι, επισημαίνονται και πάλι ότι πρέπει να χρησιμοποιηθεί η μέχρι τώρα λύση.

4.2 Έλεγχος λαθών με αναγνώριση προτύπων

Η δεύτερη διδακτική ώρα ξεκινά ζητώντας από τους μαθητές να φορτώσουν στο προγραμματιστικό τους περιβάλλον τις λύσεις τους τις οποίες και επιθεωρούμε. Η εμπειρία μας δείχνει ότι οι περισσότεροι μαθητές δεν χρησιμοποιούν ότι έχει ειπωθεί στην 1η διδακτική ώρα και ουσιαστικά ξανασκέφτονται το πρόβλημα από την αρχή, δίνοντας ως λύση κάποια παραλλαγή από αυτές που αναφέραμε για τον αλγόριθμο **Αλγόριθμος 4**. Το συγκεκριμένο στάδιο είναι το πιο δύσκολο για τους διδάσκοντες, καθώς οι μαθητές έχουν την τάση να βρίσκουν αρκετά περίεργες λύσεις οι οποίες χρειάζονται χρόνο να ελεγχθούν, όπως φαίνεται στην **Εικόνα 1**:



Εικόνα 1. *Λύσεις μαθητών Γυμνασίου*

Εδώ εφαρμόζουμε το εργαλείο της αναγνώρισης προτύπων το οποίο συμβαδίζει με αρχές από την ανακαλυπτική μάθηση (Shunk, 2010) καθώς ζητούμε από τους μαθητές να προσπαθήσουν να ελέγξουν την ορθότητα της λύσης τους, επισημαίνοντας ότι πρέπει να ψάξουν για ασυνήθιστες περιπτώσεις. Μετά από επαρκή χρόνο αναζήτησης και αν κανένα ζευγάρι δεν την έχει βρει μια περίπτωση επισημαίνουμε την περίπτωση της ισότητας δύο από τους τρεις αριθμούς. Στο ενδεχόμενο μιας περίεργης λύσης, η οποία δουλεύει σε όλες τις περιπτώσεις, αφού συγχαρούμε τους μαθητές, θέτουμε τον προβληματισμό του κατά πόσο εύκολη είναι η επέκταση της προσέγγισης τους σε 4 και παραπάνω αριθμούς.

Στη συνέχεια, ανακαλούμε την λύση για την περίπτωση των δύο αριθμών που περιγράψαμε την πρώτη διδακτική ώρα και καθοδηγούμε τους μαθητές να την επεκτείνουν. Για να γίνει αυτό επισημαίνουμε ότι ο τρόπος για να φθάσει ο μαθητής από την δεύτερη στην τρίτη τιμή (**Αλγόριθμος 1**, γραμμές 7-11) είναι παρόμοιος με τον τρόπο που έφθασε από την πρώτη στη δεύτερη (**Αλγόριθμος 1**, γραμμές 3-6).

Αφού οι μαθητές συντάξουν τις εντολές, τους προτρέπουμε να τις δοκιμάσουν πάλι, ελέγχοντας και τις τιμές που παρουσίασαν πρόβλημα νωρίτερα. Τέλος για να ελέγξουμε την κατανόηση, ζητούμε ως μια γρήγορη άσκηση να τροποποιήσουν το πρόγραμμά τους για να βρίσκει το μέγιστο από τέσσερις αριθμούς.

4.3 Γενίκευση

Ανάλογα με το επίπεδο, τις γνώσεις και την ηλικία των μαθητών μπορούμε να τους κινήσουμε το ενδιαφέρον ώστε να προσπαθήσουν να βρουν τον μέγιστο από περισσότερους αριθμούς. Επιλέγουμε για πλήθος αρχικά μία τιμή που κάνει δύσκολη την αντιγραφή των εντολών, αλλά εύκολη την εισαγωγή τους π.χ. το 10. Στο τέλος της δεύτερης διδακτικής ώρας ζητούμε λοιπόν από τους μαθητές ως άσκηση για το επόμενο μάθημα, να τροποποιήσουν τον **Αλγόριθμος 2**, χρησιμοποιώντας δομή επανάληψης. Αυτό θα τους αναγκάσει να αναλογιστούν το πλήθος των μεταβλητών που χρειάζεται να χρησιμοποιήσουν. Τους βοηθούμε, αν κρίνουμε απαραίτητο, ρωτώντας πόσες μεταβλητές χρειάζονται σε κάθε σύγκριση που γίνεται.

Την τρίτη διδακτική ώρα συζητούμε τις λύσεις των μαθητών. Σε προχωρημένα τμήματα γυμνασίου έχουμε φθάσει μέχρι αυτό το σημείο. Φυσικά σε τμήματα της Β Λυκείου πρέπει να εξεταστούν και παραλλαγές με άγνωστο πλήθος αριθμών, να γίνει κατάλληλη συζήτηση για διαφορετικούς τρόπους αρχικοποίησης των μεταβλητών και τις συνέπειές τους, αλλά και για την χρήση διαφορετικών εντολών επανάληψης.

5. Συμπεράσματα

Στην εργασία αυτή χρησιμοποιήσαμε αρχές της υπολογιστικής σκέψης στα πλαίσια διδασκαλίας ενός τυπικού αλγόριθμου που συναντούμε στον προγραμματισμό και όχι για κάποιο άλλο αντικείμενο εκτός της πληροφορικής. Η μεθοδολογία αυτή μας επέτρεψε να αξιοποιήσουμε ένα κομμάτι της ύλης, που συνήθως γίνεται αντιληπτό μόνο σε θεωρητικό επίπεδο, ώστε να κατασκευάσουν οι μαθητές κώδικα που λύνει κάποιο υπαρκτό πρόβλημα. Η αξία της συγκεκριμένης προσέγγισης φαίνεται όταν συγκρίνονται τα αποτελέσματά της με άλλες εναλλακτικές λύσεις που υπάρχουν για το ίδιο πρόβλημα. Ενσωματώσαμε τη σύγκριση αυτή στην προσέγγισή μας ώστε να γίνουν τα πλεονεκτήματά της σαφή και στους ίδιους τους μαθητές με βιωματικό τρόπο χωρίς να τους τα ‘επιβάλλει’ κάποιος θεωρητικά.

Αναφορές

Shunk, H. D. (2010). *Θεωρίες Μάθησης.* Μεταίχμιο.

Spolsky, J. (2002, November 2). *The Law of Leaky Abstractions.* Ανάκτηση από Joel on Software: www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/

Wing, J. M. (2006, March). Computational Thinking. *Communications Of The ACM, 49*(2), 33-35.

Αδαμόπουλος, Ν. (2005). Χρήση Αναλογιών και Μεταφορών στη Διδασκαλία του Μαθήματος 'Μετάδοση Δεδομένων & Δίκτυα Υπολογιστών': Μια μελέτη Περίπτωσης. *3ο Πανελλήνιο Συνέδριο "Διδακτική της Πληροφορικής".* Κόρινθος.

Αράπογλου, Α., Μαβόγλου, Χ., Οικονομάκος, Η., & Φύτρος, Κ. (2006). *Πληροφορική Α', Β', Γ' Γυμνασίου.* Αθήνα: Ο.Ε.Δ.Β.

Βακάλη, Α., Γιαννόπουλος, Η., Ιωαννίδης, Ν., Κοίλιας, Χ., Μάλαμας, Κ., Μανωλόπουλος, Ι., & Πολίτης, Π. (1999). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον.* Αθήνα: ΟΕΔΒ - ΙΕΠ.

Δουκάκης, Σ., Δουληγέρης, Χ., Καρβουνίδης, Θ., Κοίλιας, Χ., & Πέρδος, Α. (2014). *Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ.* Αθήνα: Ινστιτούτο Τεχνολογίας Υπολογιστών και Εκδόσεων «ΔΙΟΦΑΝΤΟΣ».

Ζερβουδάκη , Ε., & Παπαδάκης, Σ. (2019). Χρήση Εκπαιδευτικής Ρομποτικής και Ανάπτυξη Υπολογιστικής Σκέψης στην Προσχολική και Πρωτοσχολική Ηλικία. *10th Conference on Informatics in Education 2018* (σσ. 389-398). Εκδόσεις Νέων Τεχνολογιών.

Κοταρίνου, Π., Κουλέτση, Ε., Πλιάκου, Μ., Συριόπουλος, Σ., & Χούπη, Μ. (2018). Hobbits και Orcs: Διασχίζοντας έναν ποταμό με τους ήρωες του Tolkien. *10th Conference on Informatics in Education 2018* (σσ. 262-277). Εκδόσεις Νέων Τεχνολογιών.

Κοτίνη, Ι., & Τζελέπη, Σ. (2016). Η μεθοδολογία Agile και η εφαρμογή της στην μαθησιακή διαδικασία ενισχύουν την Υπολογιστική Σκέψη. *8th Conference on Informatics in Education* (σσ. 212-222). ΕΠΥ.

Νείρος, Α., & Ζάχαρης, Κ. (2018). Δραστηριότητες Υπολογιστικής Σκέψης. *10th Conference on Informatics in Education 2018* (σσ. 399-409). Εκδόσεις Νέων Τεχνολογιών.

Abstract

We present a teaching proposal for the computation of the maximum value of an element in a set using computational thinking concepts. The problem selected is simple enough so that it can be understood without the need for intricate examples that might divert students' attention. Furthermore, it can illustrate, in a straightforward way, important computational thinking principles, applicable in all phases ranging from problem analysis to coding.

Λέξεις κλειδιά: Minimum\Maximum value computation, computational thinking, problem analysis, recursive thinking, pattern matching