10ο ΜΑΘΗΤΙΚΟ ΣΥΝΕΔΡΙΟ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
2009-2018 Δέκα Χρόνια Μαθητικής Ψηφιακής Καινοτομίας

Bubble Hack
Οπτικοποίηση του αλγορίθμου ταξινόμησης Bubble Sort στο Scratch

[1]Καλαμποκάς Ιάσων, [2]Καραστάθη Μαρία, [3]Καραστάθη Ουρανία, [4]Χαλβατσιώτης Γεώργιος, [5]Κωνσταντίνου Ζωή, [6]Καρόγλου Νικόλαος, [7]Σουργκούνης Δημοσθένης, [8]Χατζόπουλος Ιωάννης

1..8 Μαθητές Γ Γυμνασίου, Ελληνικό Κολλέγιο Θεσσαλονίκης

Κωνσταντίνος Παρασκευόπουλος
Καθηγητής Πληροφορικής (ΠΕ19)
Ελληνικό Κολλέγιο Θεσσαλονίκης
diparaske@gmail.com

Περίληψη

Η μεγάλη χρησιμότητα των αλγορίθμων ταξινόμησης αποδεικνύεται σήμερα στην πράξη σε περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, όπως σε βιβλιοθηκονομικά συστήματα, λεξικά, τηλεφωνικούς καταλόγους και γενικά όπου γίνεται αναζήτηση αποθηκευμένων δεδομένων. Αναγνωρίζοντας αυτή την μεγάλη σημασία των αλγορίθμων ταξινόμησης, θελήσαμε να οπτικοποιήσουμε και να προσομοιώσουμε τη λειτουργία του αλγορίθμου ταξινόμησης Bubble Sort που διδαχτήκαμε φέτος στο μάθημα της πληροφορικής ώστε να μπορέσουμε να τον κατανοήσουμε καλύτερα αλλά και για να εξελίξουμε περισσότερο τις προγραμματιστικές μας ικανότητες. Δημιουργήσαμε ένα πρόγραμμα με γραφικό περιβάλλον το οποίο μπορεί να ταξινομήσει ακολουθίες αριθμητικών δεδομένων επιδεικνύοντας παράλληλα τον τρόπο λειτουργίας του αλγορίθμου. Ο προγραμματισμός του λογισμικού έγινε στο εκπαιδευτικό περιβάλλον οπτικού προγραμματισμού Scratch.

Λέξεις κλειδιά: *Bubble Sort, οπτικοποίηση, προσομοίωση, Scratch.*

1. Εισαγωγή

Η τακτοποίηση των κόμβων μιας δομής με μία ιδιαίτερη σειρά είναι μία πολύ σημαντική λειτουργία που ονομάζεται ταξινόμηση (sorting) ή διάταξη (ordering). Η ταξινόμηση αποτελεί μια σημαντική δραστηριότητα η οποία συναντάται, στην επιστήμη, στην τεχνολογία αλλά και στην καθημερινή ζωή των ανθρώπων. Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, όπως σε βιβλιοθηκονομικά συστήματα, λεξικά, τηλεφωνικούς καταλόγους και γενικά παντού όπου γίνεται αναζήτηση αποθηκευμένων αντικειμένων. Ο βασικός σκοπός, δηλαδή, της ταξινόμησης είναι να διευκολυνθεί μετέπειτα η αναζήτηση των ζητούμενων πληροφοριών. Η μεγάλη σημασία της ταξινόμησης επέβαλε την αυτοματοποίησή της μέσα από την κατασκευή αλγορίθμων εκτελέσιμων από υπολογιστές. Σχετικά απλοί αλγόριθμοι είναι η ταξινόμηση με επιλογή και η ταξινόμηση με παρεμβολή. Ο πιο γρήγορος αλγόριθμος ταξινόμησης είναι η "γρήγορη ταξινόμηση" (quicksort), ενώ η ταξινόμηση ευθείας ανταλλαγής ή αλλιώς φυσαλίδας (bubble sort) είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης. Στην παρούσα εργασία δημιουργήσαμε ένα πρωτότυπο λογισμικό προσομοίωσης του αλγορίθμου φυσαλίδας ώστε να βοηθήσουμε τους συμμαθητές μας στην καλύτερη κατανόηση της λειτουργίας του. Για τον προγραμματισμό του προσομοιωτή, χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Scratch. Η γλώσσα Scratch είναι μια σχετικά καινούργια γλώσσα προγραμματισμού με γραφικό περιβάλλον με την οποία μπορούμε να κατασκευάσουμε εφαρμογές εύκολα, γρήγορα και διασκεδαστικά. Προτιμήθηκε αφού είναι μια ιδανική γλώσσα για αρχάριους προγραμματιστές σχεδιασμένη ειδικά για την εκπαίδευση.

2. Μελέτη του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής (Bubble Sort)

Ένας από τους πιο ευρέως χρησιμοποιημένους αλγορίθμους αναζήτησης είναι ο αλγόριθμος φυσαλίδας (bubble sort), ο οποίος είναι ένας απλός και ευθύς αλγόριθμος για την ταξινόμηση δεδομένων. Η ταξινόμηση φυσαλίδας είναι γνωστή για τους αργούς της χρόνους, είναι όμως η πιο απλή από τους υπόλοιπους αλγόριθμους και γι' αυτό το λόγο είναι η πρώτη μέθοδος ταξινόμησης που μαθαίνουν οι περισσότεροι.

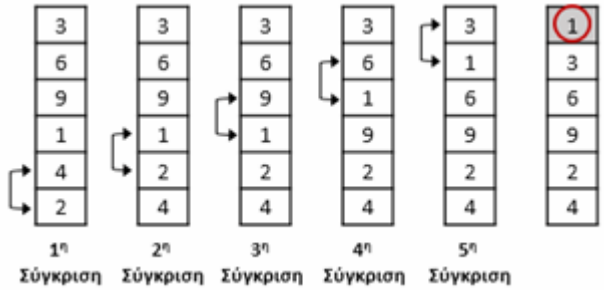
2.1. Περιγραφή του αλγορίθμου για μονοδιάστατους πίνακες

Η μέθοδος της ταξινόμησης της ευθείας ανταλλαγής βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων μέχρι να διαταχθούν όλα τα στοιχεία. Σύμφωνα με τη μέθοδο, κάθε φορά γίνονται διαδοχικές προσπελάσεις (περάσματα) στον πίνακα. Σε κάθε προσπέλαση διατρέχονται όλα τα στοιχεία του πίνακα σε αντίθετη φορά (ξεκινώντας από το τελευταίο στοιχείο) και πραγματοποιούνται συγκρίσεις διαδοχικών θέσεων. Αν το στοιχείο της μεγαλύτερης θέσης είναι μικρότερο από το στοιχείο της προηγούμενής της, τότε τα στοιχεία πρέπει να αντιμετατεθούν, δηλαδή να αλλάξουν θέσεις. Στο τέλος κάθε προσπέλασης η μικρότερη τιμή του πίνακα μετακινείται στις μπροστινές θέσεις του πίνακα. Αν ο πίνακας θεωρηθεί σε κατακόρυφη θέση αντί σε οριζόντια και το περιεχόμενό του θεωρηθεί - επιστρατεύοντας αρκετή φαντασία - ως φυσαλίδες σε δεξαμενή νερού, τότε κάθε προσπέλαση του πίνακα έχει ως αποτέλεσμα την άνοδο της ελαφρύτερης (μικρότερης τιμής του πίνακα) από τις φυσαλίδες που δεν έχουν ταξινομηθεί ακόμα στο κατάλληλο επίπεδο, δηλαδή στην κατάλληλη θέση στον πίνακα.

Ας μελετήσουμε, λοιπόν, βήμα προς βήμα τον αλγόριθμο της φυσαλίδας εφαρμόζοντάς τον στον παρακάτω μονοδιάστατο πίνακα Α:

	A
1	3
2	6
3	9
4	1
5	4
6	2

Κατά την πρώτη προσπέλαση γίνονται οι ακόλουθες συγκρίσεις:



1η Προσπέλαση

Σύγκριση 1η Αρχικά συγκρίνονται τα δύο στοιχεία που βρίσκονται στις δύο τελευταίες θέσεις, δηλαδή το 4 και το 2. Επειδή το στοιχείο στη μεγαλύτερη θέση (δηλαδή στην τελευταία) είναι μικρότερο από αυτό της προηγούμενής της (δηλαδή της προτελευταίας), αυτά αλλάζουν θέση (αντιμετατίθενται).

Σύγκριση 2η Στη συνέχεια έχουμε τη σύγκριση των στοιχείων που βρίσκονται στην 5η και την 4η θέση, δηλαδή του 2 και του 1 αντίστοιχα. Αυτά δεν αντιμετατίθενται γιατί μεταξύ τους έχουν τη σωστή διάταξη.

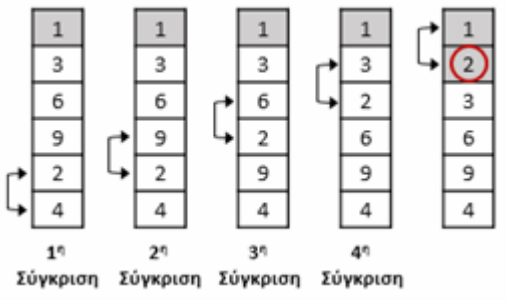
Σύγκριση 3η Η σύγκριση συνεχίζεται με τις θέσεις 4 και 3, δηλαδή με τα στοιχεία 1 και 9. Το 1 βρίσκεται χαμηλότερα από το 9, οπότε θα πρέπει να ανταλλάξουν θέσεις.

Σύγκριση 4η Με ανάλογο τρόπο συγκρίνονται τα στοιχεία των θέσεων 3 και 2, δηλαδή το 1 και το 6, οπότε και αυτά πρέπει να ανταλλάξουν θέσεις.

Σύγκριση 5η Η 1η προσπέλαση ολοκληρώνεται με τη σύγκριση των στοιχείων στις θέσεις 2 και 1, δηλαδή με τις τιμές 1 και 3, όπου και αυτές πρέπει να ανταλλάξουν θέσεις. Όταν ολοκληρωθεί η 1η προσπέλαση του πίνακα, η μικρότερη τιμή του βρίσκεται πλέον στην κορυφή, δηλαδή στη σωστή της θέση. Από εκεί και πέρα, το στοιχείο της συγκεκριμένης θέσης δεν θα συμμετάσχει ξανά σε καμία από τις συγκρίσεις των επόμενων προσπελάσεων.

Κατά τη δεύτερη προσπέλαση η διαδικασία είναι όμοια με αυτήν της πρώτης προσπέλασης, μόνο που τώρα η διαδικασία θα σταματήσει όταν συγκριθεί το στοιχείο της 3ης θέσης με αυτό της 2ης. Δηλαδή έχει μία σύγκριση

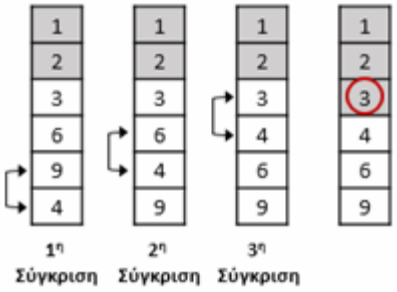
λιγότερη από ό,τι η πρώτη προσπέλαση. Με την ολοκλήρωση αυτής της προσπέλασης ένα ακόμα στοιχείο βρίσκεται πλέον στη σωστή του θέση (2η θέση) και δεν θα συμμετάσχει σε καμία σύγκριση στις υπόλοιπες προσπελάσεις.



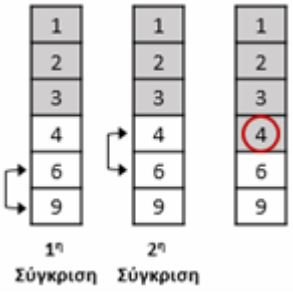
2η Προσπέλαση

Ανάλογη διαδικασία ακολουθούμε και στις υπόλοιπες προσπελάσεις οι οποίες παρουσιάζονται στη συνέχεια, όπου σε καθεμία από αυτές ένα νέο στοιχείο μεταβιβάζεται στη σωστή του θέση.

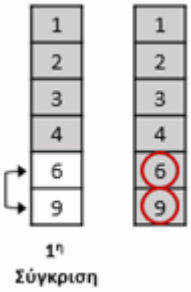
3η Προσπέλαση



4η Προσπέλαση



5η Προσπέλαση



Η 5η προσπέλαση είναι η τελευταία προσπέλαση για έναν πίνακα 6 θέσεων. Στη προσπέλαση αυτή γίνεται μόνο μία σύγκριση μεταξύ των στοιχείων των δύο τελευταίων θέσεων και για τον συγκεκριμένο πίνακα αυτά δεν πρέπει να ανταλλάξουν θέσεις.

2.2. Υλοποίηση του αλγορίθμου με ψευδογλώσσα

Κατά την πρώτη προσπέλαση γίνονται διαδοχικές συγκρίσεις ανάμεσα σε γειτονικά στοιχεία του πίνακα. Ας υποθέσουμε ότι βρισκόμαστε στη θέση j και θέλουμε να συγκρίνουμε το συγκεκριμένο στοιχείο με το στοιχείο που βρίσκεται στην προηγούμενη θέση, δηλαδή στη j-1. Για να γίνει η αντιμετάθεση των στοιχείων, θα πρέπει το περιεχόμενο της θέσης j να είναι μικρότερο από το περιεχόμενο της θέσης j-1 (για ταξινόμηση σε αύξουσα σειρά). Έτσι για τις θέσεις j και j-1 έχουμε το ακόλουθο τμήμα αλγορίθμου:

```
Αν A[j] < A[j-1] τότε
    temp ← A[j]
    A[j] ← A[j-1]
    A[j-1] ← temp
Τέλος_αν
```

Αν το στοιχείο της θέσης j είναι μικρότερο από αυτό της θέσης j-1, τότε οι θέσεις να ανταλλάξουν περιεχόμενα.

Κατά την πρώτη προσπέλαση του πίνακα γίνεται αρχικά σύγκριση της θέσης N με το περιεχόμενο της θέσης N-1, δηλαδή το j αρχικά έχει την τιμή N. Στη συνέχεια γίνεται σύγκριση του περιεχομένου της θέσης N-1 με αυτό της θέσης N-2 (δηλαδή το j=N-1) και συνεχίζουμε ώσπου να γίνει και η σύγκριση του περιεχομένου της θέσης 2 με της θέσης 1 (δηλαδή το j=2). Άρα το παραπάνω τμήμα πρέπει να εκτελεστεί για κάθε j από τη N-οστή μέχρι και τη 2η. Επομένως για την πρώτη προσπέλαση έχουμε το ακόλουθο τμήμα:

```
Για j από N μέχρι 2 με_βήμα -1
    Αν A[j] < A[j-1] τότε
        temp ← A[j]
        A[j] ← A[j-1]
        A[j-1] ← temp
    Τέλος_αν
Τέλος επανάληψης
```

Για κάθε θέση j από τη N-οστή μέχρι και τη 2η, αν το στοιχείο της θέσης j είναι μικρότερο από αυτό της θέσης j-1, τότε οι θέσεις να ανταλλάξουν περιεχόμενα.

Κατά τη δεύτερη προσπέλαση πρέπει να εκτελεστεί το ίδιο ακριβώς τμήμα αλγορίθμου, αλλά, επειδή η τελευταία σύγκριση πρέπει να είναι μεταξύ των θέσεων 3ης και 2ης (και όχι μεταξύ 2ης και 1ης), η τελική τιμή του j πρέπει να είναι το 3 και όχι το 2 που ήταν στην προηγούμενη προσπέλαση. Άρα για τη δεύτερη προσπέλαση πρέπει να

εκτελεστεί ακριβώς το ίδιο τμήμα αλγορίθμου της προηγούμενης προσπέλασης μόνο που αντί η τελική τιμή της για…από…μέχρι να είναι το 2 πρέπει να είναι το 3 (για j από N μέχρι 3 με_βήμα -1).

Αντίστοιχα για την τρίτη προσπέλαση, η μόνη διαφορά είναι ότι η τελική τιμή του μετρητή j πρέπει να είναι το 4 κ.ο.κ. Για την τελευταία προσπέλαση, όπου έχουμε μία μόνο σύγκριση της θέσης N με τη θέση N-1, η τελική τιμή του μετρητή j πρέπει να είναι η N (για j από N μέχρι N με_βήμα -1).

Άρα συνοπτικά μπορούμε να πούμε ότι η πρώτη γραμμή του τμήματος αλγορίθμου κάθε προσπέλασης είναι η ακόλουθη:

Για την πρώτη προσπέλαση: **Για** j **από** N **μέχρι** 2 **με_βήμα** -1
Για τη δεύτερη προσπέλαση: **Για** j **από** N **μέχρι** 3 **με_βήμα** -1
.....
Για την τελευταία προσπέλαση: **Για** j **από** N **μέχρι** N **με_βήμα** -1

Έτσι, λοιπόν, αν επιτρέψουμε σε μια μεταβλητή i να πάρει διαδοχικά για καθεμία από τις προσπελάσεις τις τελικές τιμές του j (που βρίσκονται μέσα σε κύκλο), αποφεύγουμε να γράψουμε ένα ξεχωριστό τμήμα αλγορίθμου για καθεμία από αυτές, και έτσι προκύπτει ο ολοκληρωμένος πλέον αλγόριθμος της φυσαλίδας:

```
Αλγόριθμος Φυσαλίδα
    Δεδομένα //A, N//
    Για i από 2 μέχρι N
        Για j από N μέχρι i με_βήμα -1
            Αν A[j] < A[j-1] τότε
                temp ← A[j]
                A[j] ← A[j-1]
                A[j-1] ← temp
            Τέλος_αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
    Αποτελέσματα //A//
Τέλος Φυσαλίδα
```

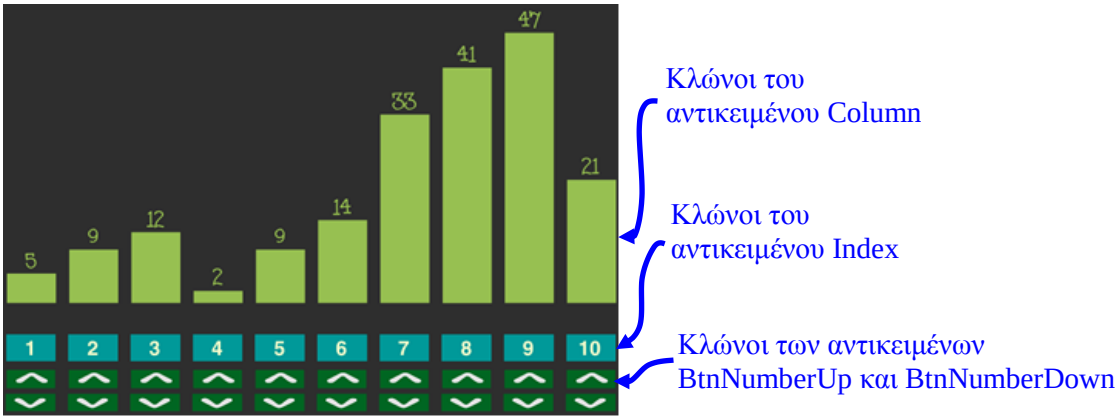
3. Ο Προγραμματισμός της εφαρμογής στη γλώσσα Scratch

Αφού κατανοήσαμε πλήρως τη λειτουργία του αλγορίθμου Bubble Sort ήρθε η ώρα να προγραμματίσουμε το λογισμικό προσομοίωσης και οπτικοποίησης στο Scratch. Πριν να ξεκινήσουμε όμως τη διαδικασία του προγραμματισμού, καταγράψαμε όλες τις **λειτουργικές απαιτήσεις** του υπό ανάπτυξη λογισμικού. Καταλήξαμε ότι ο χρήστης στην αλληλεπίδραση του με το λογισμικό θα πρέπει να έχει στη διάθεσή του συγκεκριμένες λειτουργίες μέσα από το μενού του προγράμματος. Οι λειτουργίες αυτές είναι :

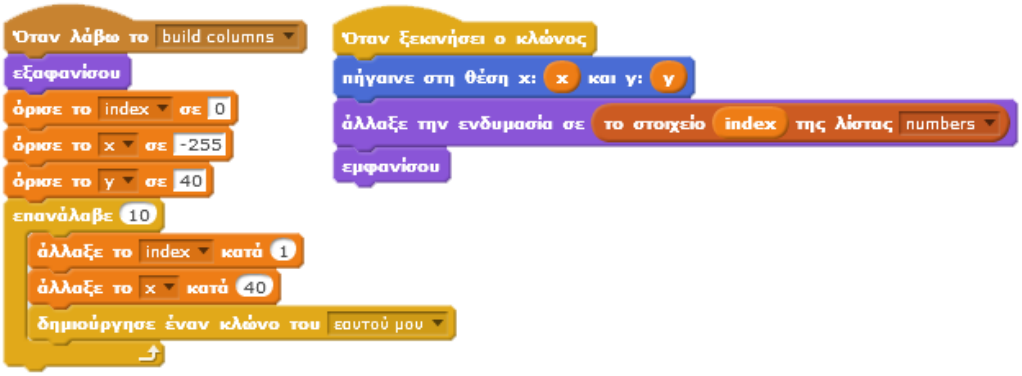
1. Δημιουργία πίνακα 10 θέσεων. Η δημιουργία του πίνακα θα πρέπει να γίνεται με το πάτημα ενός κουμπιού. Σε κάθε κελί του πίνακα θα πρέπει να καταχωρούνται τυχαίες ακέραιες τιμές. Οι τιμές αυτές θα πρέπει να ανήκουν στο διάστημα [1, 50].
2. Δυνατότητα αυξομείωσης των τιμών του πίνακα. Η αυξομείωση των τιμών του πίνακα θα γίνεται με κατάλληλα χειριστήρια.
3. Εκκίνηση του αλγορίθμου ταξινόμησης με το πάτημα ενός κουμπιού.
4. Τερματισμός του αλγορίθμου ταξινόμησης με το πάτημα ενός κουμπιού.
5. Αυξομείωση της ταχύτητας εκτέλεσης των βημάτων του αλγορίθμου με χρήση κατάλληλων χειριστηρίων.

Επίσης το λογισμικό θα πρέπει να έχει τη δυνατότητα οπτικοποίησης των τιμών του πίνακα και παρουσίασής τους με κατάλληλα γραφικά στο χρήστη. Αποφασίσαμε τις τιμές του πίνακα να τις αναπαραστήσουμε με στήλες το ύψος των οποίων θα είναι ανάλογο με τις τιμές που αναπαριστούν.

Στην ***Εικόνα 1*** φαίνεται η οπτικοποίηση του πίνακα. Οι στήλες δημιουργούνται ως κλώνοι χρησιμοποιώντας το αντικείμενο **Column** με τις εντολές που φαίνονται στην ***Εικόνα 2***. Στη βάση κάθε στήλης φαίνεται ο αριθμός της θέσης της μέσα σε ένα πλαίσιο. Τα πλαίσια με τις θέσεις των στηλών δημιουργούνται ως κλώνοι χρησιμοποιώντας το αντικείμενο **Index.** Ακριβώς κάτω από τα πλαίσια με τις θέσεις των στηλών φαίνονται τα χειριστήρια για την αυξομείωση των τιμών του πίνακα στις αντίστοιχες θέσεις. Τα χειριστήρια δημιουργούνται ως κλώνοι των αντικειμένων **BtnNumberUp** και **BtnNumberDown.**

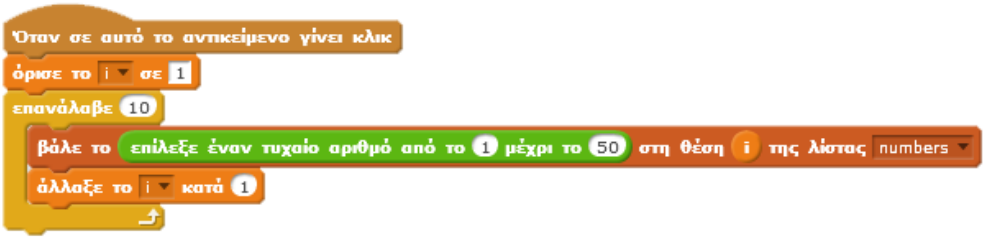


***Εικόνα 1**: Οπτικοποίηση του πίνακα και των χειριστηρίων για την αυξομείωση των τιμών.*



***Εικόνα 2**: Δημιουργία των στηλών για την γραφική αναπαράσταση του πίνακα..*

Όπως περιγράψαμε και στις λειτουργικές απαιτήσεις του λογισμικού, ένας πίνακας 10 τυχαίων ακέραιων τιμών θα μπορεί να δημιουργηθεί με το πάτημα ενός κουμπιού. Δημιουργήσαμε λοιπόν το αντικείμενο **BtnRandom** και προσθέσαμε τις εντολές που φαίνονται στην ***Εικόνα 3***.



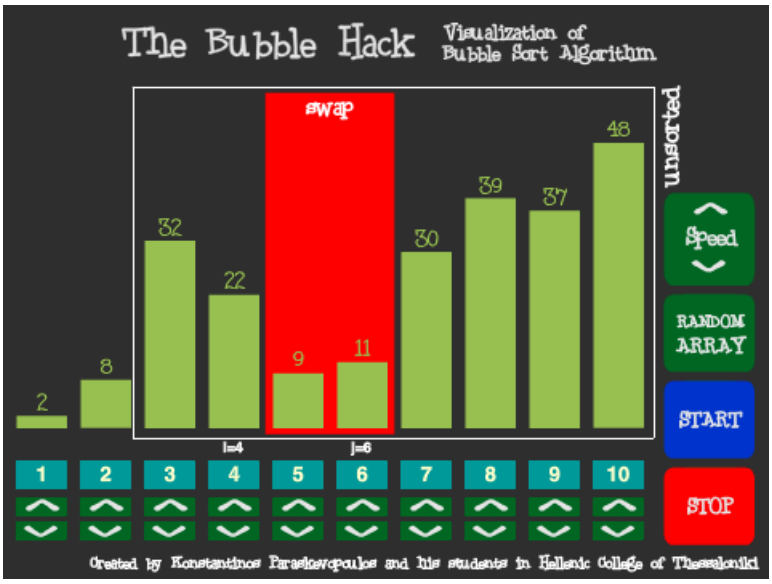
***Εικόνα 3**: Δημιουργία λίστας 10 τυχαίων ακέραιων αριθμών.*

***Εικόνα 4**: Υλοποίηση του αλγορίθμου ταξινόμησης Bubble Sort στο Scratch.*

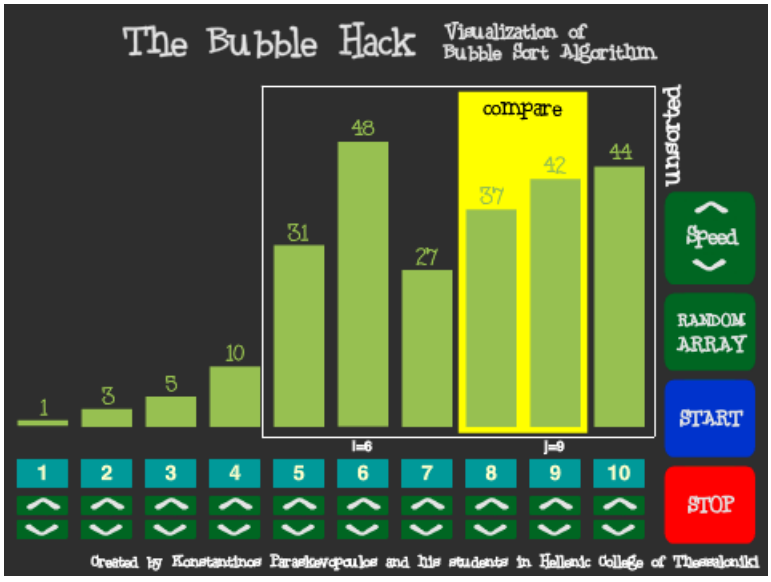
Όπως περιγράψαμε και στις λειτουργικές απαιτήσεις του λογισμικού η διαδικασία της ταξινόμησης πρέπει να ξεκινάει με το πάτημα ενός κουμπιού από τον χρήστη της εφαρμογής. Δημιουργήσαμε λοιπόν το αντικείμενο

BtnStart και προσθέσαμε εντολές για να υλοποιήσουμε τον αλγόριθμο Bubble Sort όπως φαίνονται στην ***Εικόνα 4***.

Στη συνέχεια προσθέσαμε ανάμεσα στις εντολές του αλγορίθμου, όπου χρειαζόταν, κατάλληλες εντολές για τον έλεγχο των γραφικών αλλά και της ταχύτητας εκτέλεσης των εντολών. Το τελικό αποτέλεσμα φαίνεται στην ***Εικόνα 5***. Το λογισμικό το ανεβάσαμε στον διαδικτυακό τόπο **scratch.mit.edu** και συγκεκριμένα στη διεύθυνση https://scratch.mit.edu/projects/208361972/ .



Εικόνα 5*: Το λογισμικό κατά την εκτέλεση μιας ταξινόμησης. Παρατηρήστε ότι μόλις έγινε μια ανταλλαγή (swap) δύο γειτονικών στοιχείων του πίνακα.*



Εικόνα 6*: Το λογισμικό κατά την εκτέλεση μιας ταξινόμησης. Παρατηρήστε ότι γίνεται σύγκριση (compare) δύο γειτονικών στοιχείων του πίνακα.*

Ευχαριστίες

Θα θέλαμε να εκφράσουμε τις ευχαριστίες μας στον καθηγητή πληροφορικής του Ελληνικού Κολλεγίου Θεσσαλονίκης, κ. Κωνσταντίνο Παρασκευόπουλο για την πολύτιμη καθοδήγηση του στην υλοποίηση της εργασίας.

Βιβλιογραφία

1. Παρασκευόπουλος Κ., Εισαγωγή στον Προγραμματισμό με το Scratch, Ελληνικό Κολλέγιο Θεσσαλονίκης 2017.
2. Παρασκευόπουλος Κ., Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον, Ελληνικό Κολλέγιο Θεσσαλονίκης, 2015.