**ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ**

**Πρόγραμμα Σπουδών**

**Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα**

**Θεματική Ενότητα**

**ΠΛΣ50 – Βασικές εξειδικεύσεις σε Θεωρία και Λογισμικό**

Ακαδημαϊκό Έτος: 2016-2017

**Δεύτερη Γραπτή Εργασία**

**Γλώσσα C (Β)**

Φοιτητής: Λιάγκος Αθανάσιος

Τμήμα: ΑΘΗ-2

Αριθμός Μητρώου: 123575

Καθηγητής: Μαυρομμάτης Γεώργιος

**Αθήνα Δεκέμβριος 2016**

# ΕΙΣΑΓΩΓΗ

Οι ασκήσεις έχουν λυθεί και μεταγλωττιστεί με το CLion[[1]](#footnote-1) της JetBrains σε περιβάλλον Windows 10 x64 με χρήση του gcc 5.4.0 του πακέτου Cygwin64[[2]](#footnote-2) (έκδοση 2.6.1) και CMake 3.6.2.

# ΑΣΚΗΣΗ 1

Στην άσκηση αυτή θα τροποποιήσουμε το πρόγραμμα που είχαμε φτιάξει για το 4ο ερώτημα της 1ης ΓΕ, ώστε να μπορούμε να έχουμε ονοματεπώνυμο και τηλέφωνο για κάθε επιβάτη.

Για την υλοποίηση, ορίσαμε μια δομή PASSENGER που θα έχει τα στοιχεία ενός επιβάτη και μια δομή BUSDATA η οποία θα περιέχει την πινακίδα του λεωφορείου και τον αριθμό θέσεων. Κρίθηκε σκόπιμα να φτιαχτούν δύο συναρτήσεις, οι οποίες μετατρέπουν ένα string που περιέχει το τηλέφωνο σε array 10 unsigned short και το αντίστροφο, ώστε να είναι το πρόγραμμα σύμφωνο με τα ζητούμενα της άσκησης, αλλά εύκολο στην εμφάνιση και εισαγωγή τηλεφώνου. Οι κύριες συναρτήσεις παίρνουν ως παράμετρο τον πίνακα με τα στοιχεία των επιβατών, και επιπλέον οι περισσότερες παίρνουν και ως παράμετρο τον μέγιστο αριθμό θέσεων που έχει διαβαστεί από το αρχείο

Θα αποφύγουμε περιττά σχόλια στον κώδικά, και θα επικεντρωθούμε στις βασικές τροποποιήσεις και προσθήκες.



Ορισμός του μέγιστου αριθμού θέσεων καθώς και των δομών που θα χρησιμοποιηθούν.



Αναλόγως τι μετατροπή θέλουμε να κάνουμε, μετατρέπουμε έναν πίνακα 10 unisgned short σε string, ή το ανάποδο. Με ένα for διατρέχουμε όλες τις τιμές της ανάλογης παραμέτρου, λαμβάνοντας υπόψη πως το νούμερο 0 είναι ο χαρακτήρας 48 στον πίνακα ASCII. Κάνουμε έτσι μία πρόσθεση ή αφαίρεση για να βρούμε τη σωστή τιμή, και την αποθηκεύομε στις μεταβλητές που έχουνε κληθεί με αναφορά.



Ο πίνακας αρχικοποιείται θέτοντας στο όνομα τον κενό χαρακτήρα και στη θέση την τρέχουσα θέση.



Η συνάρτηση **showMenu** εμφανίζει το μενού με τις επιλογές στον χρήστη.



Με τη συνάρτηση αυτή, διαβάζουμε όλα τα δεδομένα από το υπάρχον αρχείο. Τα μεν στοιχεία του λεωφορείου τα επιστρέφουμε ως την τιμή της συνάρτησης, ενώ τα στοιχεία του πίνακα των επιβατών που έχει περαστεί με αναφορά, συμπληρώνονται για όλες τις θέσεις που υπάρχουν στο αρχείο. Κάνοντας το fscanf για την πρώτη γραμμή των επιβατών, αν δεν υπάρχουν εγγραφές δεν θα μπει καθόλου στο while, ενώ αν υπάρχει έστω και μία θα μπει. Το while θα εκτελείται όσο υπάρχουν γραμμές στο αρχείο. Μέσα στο while μετατρέπουμε το τηλέφωνο σε πίνακα unsigned short, προσθέτουμε ένα κενό στο επώνυμο και κολλάμε στο τέλος το όνομα. Όλα αυτά αποθηκεύονται στην current, την οποία με τις σειρά της την εκχωρούμε στη θέση seat-1 του πίνακα layout.



Εδώ απλά εμφανίζουμε το μενού επιλογών.



Με τη συνάρτηση αυτή τυπώνουμε τις κενές θέσεις, ενώ κάθε 10 θέσεις τυπώνουμε και το χαρακτήρα νέας γραμμής. Για να βρούμε τις κενές θέσεις, διατρέχουμε όλον τον πίνακα και κοιτάμε αν στην πρώτη θέση του ονόματος είναι ο χαρακτήρας του κενού στοιχείου. Αν δεν υπάρχουν κενές θέσεις, εμφανίζεται ανάλογο μήνυμα.



Με τη συνάρτηση αυτή κάνουμε κράτηση μιας συγκεκριμένης θέσης. Η παράμετρος είναι η διεύθυνση της συγκεκριμένης θέσης στον πίνακα, ενώ γίνεται πρώτα έλεγχος αν είναι όντως κενή. Στη συνέχεια ο χρήστης δίνει τα στοιχεία, μετατρέπουμε πάλι το τηλέφωνο από string σε πίνακα unsigned short και πλέον η συγκεκριμένη θέση έχει τα νέα στοιχεία.



Εδώ ελέγχουμε αν μια θέση είναι κρατημένη βάσει ονόματος ή τηλεφώνου. Εμφανίζουμε τις επιλογές στο χρήστη και ανάλογα αν έχει ζητήσει αναζήτηση με όνομα ή τηλέφωνο, διαβάζει τα αντίστοιχα πεδία.



Ψάχνουμε στη συνέχεια όλες τις θέσεις, μετατρέπουμε το τηλέφωνο της θέσης σε string και μετά με την strcmp ελέχγουμε, και ανάλογα αν η αναζήτηση έγινε με όνομα ή τηλέφωνο, ελέγχουμε αν επιστρέφει 0, δηλαδή να είναι ταυτόσημα τα στοιχεία. Αν βρεθεί ταύτιση τυπώνονται τα στοιχεία και θα μπορούσε ο βρόγχος να σταμτήσει εκεί με break. Κρίθηκε σκόπιμο όμως να συνεχίσει, ώστε αν υπάρχει άλλη θέση με ίδιο όνομα ή τηλέφωνο να την εμφανίσει και αυτή.



Εδώ ακυρώνουμε μια θέση θέτοντας στον πρώτη θέση του ονόματος τον κενό χαρακτήρα, ενώ αν η θέση δεν είναι κρατημένη εμφανίζουμε μήνυμα.



Εδώ διατρέχοντας όλον τον πίνακα, εμφανίζουμε τις κρατημένες θέσεις, αφού έχουμε πρώτα μετατρέψει το τηλέφωνο, καθώς και το σύνολό τους. Αν δεν υπάρχουν κρατημένες θέσεις, εμφανίζεται ανάλογο μήνυμα.



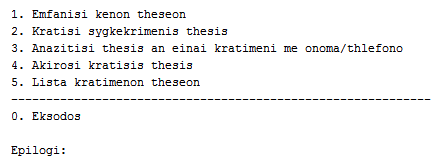
Κατά την έξοδο του προγράμματος εκτελείται αυτή η συνάρτηση η οποία γράφει το νέο αρχείο. Από την busdata παίρνουμε την πινακίδα και τις θέσεις του λεοφωρείου, ενώ μετά διατρέχοντας τον πίνακα με τους επιβάτες, γράφουμε τα στοιχεία τους στο αρχείο, αν βέβαια υπάρχει κράτηση.



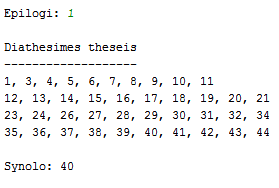
Η **main** τέλος, περιέχει τους βασικούς ορισμούς των μεταβλητών του προγράμματος, κάνει τους βασικούς ελέγχους ορθότητας των παραμέτρων και διαχειρίζεται το μενού οδηγώντας τη ροή του προγράμματος στις κατάλληλες συναρτήσεις. Ορίζουμε έναν πίνακα **MAXSEATS** θέσεων τύπου **PASSENGER** για τα στοιχεία των επιβατών, και μία μεταβλητή τύπου **BUSDATA** για τα στοιχεία του λεωφορείου. Ορίζουμε 1 μεταβλητή τύπου **char** για την επιλογή του μενού και μία **unsignen int** για τη θέση που θα δώσει ο χρήστης, ανάλογα την επιλογή του. Αρχικά μηδενίζουμε τον πίνακα τον επιβατών, και στη συνέχεια με την **getDataFromFile** διαβάζουμε τα δεδομένα του αρχείου, ενώ η πινακίδα και ο αριθμός θέσεων είναι αποθηκευμένα κατά την επιστροφή στην busdata. Μετά, σε έναν βρόγχο **do..while** εμφανίζουμε το μενού και διαβάζουμε την επιλογή του χρήστη στη μεταβλητή **option**, όσο (while) η επιλογή του χρήστη δεν είναι 0 (=έξοδος). Με τη χρήση της **switch** ελέγχουμε τις επιλογές του χρήστη, και καλούμε κάθε φορά την αντίστοιχη συνάρτηση με τις αντίστοιχες παραμέτρους. Σε περίπτωση που ο χρήστης επιλέξει 0, πριν την έξοδο του προγράμματος καλούμε την **commitLayout** η οποία γράφει το αρχείο με τα νέα δεδομένα. Σε όλες τις περιπτώσεις που απαιτείται ο χρήστης να δώσει αριθμό θέσης, γίνεται πρώτα έλεγχος αν η θέση είναι μέσα στο κατάλληλο εύρος **1..MAXEATS**.

Ακολουθούν screenshots από την εκτέλεση του προγράμματος. Το αρχείο **bus.txt** έχει τη γραμμή **ΝΒΑ6677 45**, άρα οι θέσεις είναι 45.

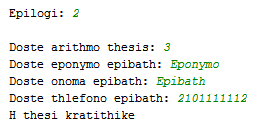
**Screenshot 1.1 – Μενού επιλογών**



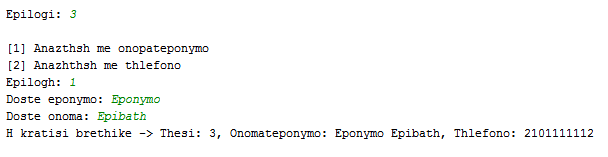
**Screenshot 1.2 – Κενές θέσεις (5 θέσεις κρατημένες)**



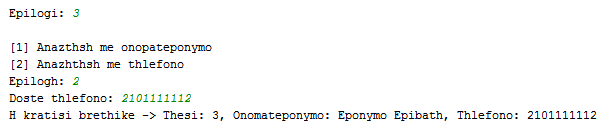
**Screenshot 1.3 – Κράτηση θέσης 3**



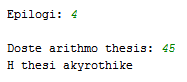
**Screenshot 1.4 – Αναζήτηση θέσης 3 με όνομα**



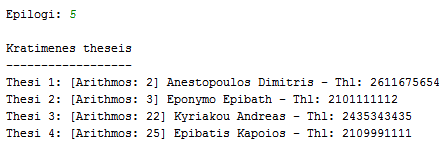
**Screenshot 1.4 – Αναζήτηση θέσης 3 με τηλέφωνο**



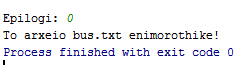
**Screenshot 1.5 – Ακύρωση θέση 45 που είχε διαβαστεί από το αρχείο**

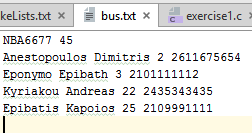


**Screenshot 1.6 – Εμφάνιση κρατημένων θέσεων**



**Screenshot 1.7 – Έξοδος και ενημέρωση αρχείου**





# ΑΣΚΗΣΗ 2

Στην άσκηση αυτή θα τροποποιήσουμε το πρόγραμμα που είχαμε φτιάξει στο προηγούμενο ερώτημα, έτσι ώστε ο πίνακας layout να ορίζεται δυναμικά (σύμφωνα με τις θέσεις που είναι στο αρχείο), καθώς και να υπάρχει δυνατότητα ταξινόμησης με όνομα και αριθμό θέσης. Πριν την έξοδο, ο πίνακας θα ταξινομείται κατά όνομα, θα μετατρέπεται σε λίστα και θα αποθηκεύεται ταξινομημένος στο αρχείο.

Για την υλοποίηση, ορίσαμε μια δομή PASSENGERlist παρόμοια με την PASSENGER, με τη διαφορά πως υπάρχει το στοιχείο **next** το οποίο θα περιέχει τη διεύθυνση του επόμενου στοιχείου στη λίστα.

Θα αποφύγουμε περιττά σχόλια στον κώδικά, και θα επικεντρωθούμε στις βασικές τροποποιήσεις και προσθήκες, σε σχέση με το προηγούμενο ερώτημα.



Εφόσον το πλήθος των στοιχείων του πίνακα θα ορίζεται δυναμικά, η **define MAXSEATS** δεν χρειάζεται. Η PASSENGERlist είναι η δομή που θα χρησιμοποιηθεί για τη μετατροπή του πίνακα σε λίστα.



Εδώ έχουμε αφαιρέσει από την **getDataFromFile** τη λειτουργεία που διαβάζει την πινακίδα και τον αριθμό θέσεων και την έχουμε ξεχωριστή συνάρτηση, διότι στην **getDataFromFile** πρέπει να ξέρουμε ήδη τον αριθμό θέσεων του λεωφορείου ώστε να έχουμε κάνει την αρχικοποίηση του πίνακα με το σωστό πλήθος θέσεων.



Στην **getDataFromFile** έχουμε προσθέσει το παραπάνω αμέσως μετά το άνοιγμα, ώστε να «προσπεράσουμε» την πρώτη γραμμή που περιέχει τα στοιχεία του λεωφορείου και να διαβάσουμε μόνο τα στοιχεία των επιβατών. Η συνέχεια είναι ίδια όπως το προηγούμενο ερώτημα, με την εξαίρεση πως πλέον δεν υπάρχει ανάγκη να επιστρέψει κάτι η συνάρτηση αυτή.

Οι συναρτήσεις **showBlanks, reserveSpecific, checkReserved** και **cancelSpecific** παραμένουν ακριβώς ίδιες όπως στο προηγούμενο ερώτημα (επιλογές 1, 2, 3 και 4).



Την **quicksort** τη χρησιμοποιύμε για την ταξινόμηση στην επιλογή 5 και στην έξδοδο. Η συνάρτηση αυτή, η οποία στην παράμετρο **type** παίρνει την τιμή 1 για ταξινόμηση με όνομα και 2 για ταξινόμηση με θέση, διατρέχει όλον τον πίνακα εκτός της τελευταίας θέσης, και με τη βοήθεια της **strcmp** ελέγχει αν η τρέχουσα θέση με την επόμενη είναι στη σωστή σειρά αλφαβητικά, ελέχγοντας αν το αποτέλεσμα είναι θετικό. Αν είναι θετικό, σημαίνει πως η τρέχουσα θέση θα πρέπει να είναι μετά την επόμενη, άρα πρέπει να γίνει ανταλλαγή θέσεων μεταξύ τους στον πίνακα. Το ίδιο ισχύει και με τον αριθμό θέσης, όπου απλά ελέγχεται ο αριθμός της μίας με την άλλη. Η συνάρτηση **swapPassengers** κάνει ακριβώς αυτή την εναλλαγή των θέσεων. Στη συνέχεια η **quicksort** καλεί τον εαυτό της, αλλά πλέον έχει γίνει ήδη μια ταξινόμηση. Αυτή η διαδικασία συνεχίζεται μέχρι την **προτελευταία** θέση του πίνακα. Αυτό συμβαίνει διότι κατά τον έλεγχο της προτελευταίας, ο έλεγχος θα γίνει σε σχέση με την τελευταία οπότε δεν υπάρχει νόημα να ελεγχθεί πάλι με τον εαυτό της. Στο τέλος της συνάρτησης ο πίνακας θα έχει την επιθυμητή ταξινόμηση.



Η συνάρτηση αυτή δημιουργεί τη λίστα από τον πίνακα layout. Η μεταβλητή **passenger** είναι ένας δείκτης που επιστρέφει τη διεύθυνση του πρώτου στοιχείου της λίστας. Η συνάρτηση παίρνει ως παράμετρο τη θέση του πίνακα, η οποία στην πρώτη κλήση έχει την τιμή 0 και σε κάθε κλήση αυξάνεται κατά 1 μέχρι να φτάσει στην τελευταία, όπου θα επιστρέψει **NULL**. Αν λοιπόν η θέση είναι κρατημένη (name[0] != ‘\0’) δεσμεύουμε μνήμη για το struct του επιβάτη, το οποίο γεμίζουμε με τα στοιχεία του. Στη θέση **next** του struct, βάζουμε τη διεύθυνση του επόμενου στοιχείου (pos+1) την οποία βρίσκουμε καλώντας την ίδια πάλι συνάρτηση με τις ανάλογες παραμέτρους. Αν η θέση είναι κενή και δεν έχουμε φτάσει στο τέλος του πίνακα (pos<maxSeats), βρίσκουμε τη διεύθυνση του επόμενου στοιχείου καλώντας πάλι την ίδια συνάρτηση. Στο τέλος όλης της διαδικασίας, η passenger θα έχει είτε την τιμή NULL αν δεν υπάρχουν κρατημένες θέσεις, είτε τη διεύθυνση του πρώτου στοιχείου της λίστας.



Εδώ έχουμε τροποποιήσει την **commitLayout** με τέτοιο τρόπο ώστε να μη διατρέχει τον πίνακα με for, αλλά τη λίστα που έχουμε φτιάξει. Η **current** θα έχει κάθε φορά τη διεύθυνση του επόμενου στοιχεί της λίστας, άρα την πρώτη φορά αν δεν υπάρχουν κρατήσεις θα έχει την τιμή NULL και δεν θα μπει στο while. Αν υπάρχει τιμή, μετατρέπουμε το τηλέφωνο σε string και γράφουμε τη γραμμή στο αρχείο μαζί με το όνομα και τον αριθμό θέσης. Στη συνέχεια η **current** παίρνει την τιμή της διεύθυνσης του επόμενου στοιχείου, και αν είναι NULL σταματάει αλλιώς συνεχίζει με τον ίδιο τρόπο.



Είμαστε πλέον στη **main()** και βλέπουμε τη διαφοροποίηση σε σχέση με το προηγούμενο ερώτημα. Διαβάζουμε πρώτα τα στοιχεία του λεωφορείου, και μετά δημιουργούμε δυναμικά τον πίνακα των επιβατών με τη **malloc** σύμφωνα με τις θέσεις που έχουμε διαβάσει. Αρχικοποιούμε στη συνέχεια τον πίνακα και διαβάζουμε τα στοιχεία των επιβατών, αν υπάρχουν, από το αρχείο.



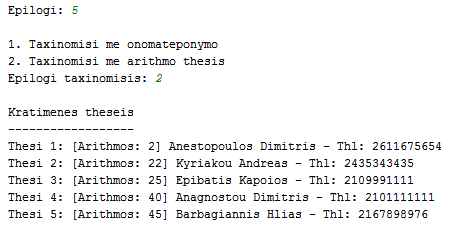
Στην επιλογή που εμφανίζουμε τις κρατημένες θέσεις, ζητάμε πρώτα από τον χρήστη να μας δώσει το είδος της ταξινόμησης που επιθυμεί. Αν η ταξινόμηση είναι κατά ονοματεπώνυμο, καλείται η **quicksort** με τύπο 1, εμφανίζεται ο πίνακας με τον γνωστό τρόπο της **showReserved** και στη συνέχεια καλείται πάλι η **quicksort** αλλά με τύπο 2 για γίνει ταξινόμηση με αριθμό θέσης. Αν η ταξινόμηση που ζητηθεί είναι με αριθμό θέσης, απλά καλούμε την **showRserved** διότι ο πίνακας είναι ήδη ταξινομημένος κατά αριθμό θέσης.



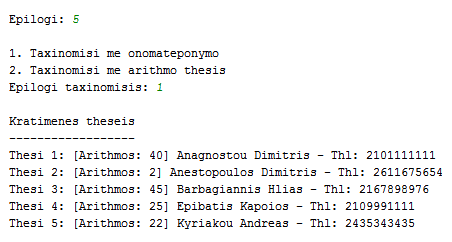
Εδώ έχουμε τροποποιήσει τη διαδικασία εξόδου, όπου αν την επιλέξει ο χρήστης κάνουμε πρώτα ταξινόμηση με ονοματεπώνυμο με την **quicksort**, και στη συνέχεια δίνουμε στην **passengerList** την διεύθυνση του πρώτου στοιχείου της λίστας. Αφού φτιαχτεί η λίστα, τη γράφουμε στο αρχείο με την **commitLayout**.

Ακολουθούν screenshots από την εκτέλεση του προγράμματος. Το αρχείο **bus.txt** έχει τη γραμμή **ΝΒΑ6677 45**, άρα οι θέσεις είναι 45, μόνο για τις περιπτώσεις που υπάρχει διαφοροποίηση σε σχέση με το προηγούμενο ερώτημα.

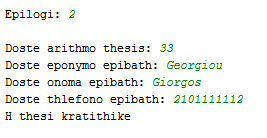
**Screenshot 2.1 – Οι θέσεις όπως έχουν διαβαστεί αρχικά (ταξινόμηση με αριθμό θέσης)**



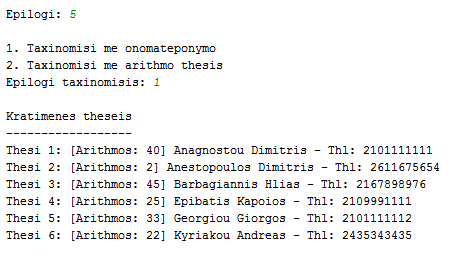
**Screenshot 2.2 – Οι θέσεις όπως έχουν διαβαστεί αρχικά (ταξινόμηση με ονομ/νυμο)**



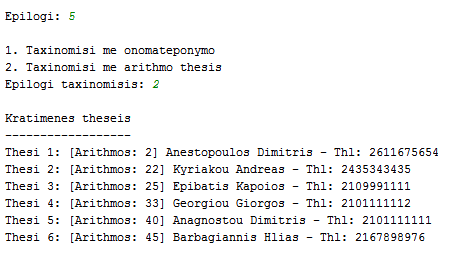
**Screenshot 2.3 – Κράτηση θέσης 33**



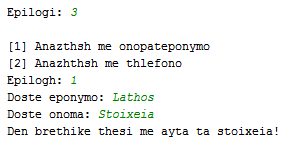
**Screenshot 2.4 – Εμφάνιση νέων θέσεων κατά ονοματεπώνυμο**



**Screenshot 2.5 – Εμφάνιση νέων θέσεων κατά αριθμό θέσης**

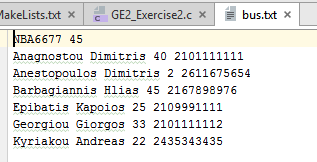


**Screenshot 2.6 – Αναζήτηση με λάθος στοιχεία**



**Screenshot 2.7 – Έξοδος, ταξινόμηση, αποθήκευση αρχείου**

C:\Users\nasos\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Screenshot_27.png



# ΑΣΚΗΣΗ 3

Στην άσκηση αυτή θα φτιάξουμε ένα απλό παιχνίδι, όπου θα υπάρχουν 3 συγκεκριμένοι παίκτες και ο κάθε παίκτης θα πρέπει να βρει συνδυασμούς καρτών επιλέγοντας ανάμεσα σε 12 κάρτες 3, οι οποίες θα έχουνε όλα τα επιμέρους στοιχεία ίδια είτε όλα διαφορετικά.



Τις δομές αυτές θα τις χρησιμοποιήσουμε τη μεν **CARD** για να αποθηκεύσουμε τους 81 πιθανούς συνδυασμούς καρτών, και τη δε **PLAYER** για να έχουμε τα στοιχεία των παικτών και το σκορ τους. Θα χρησιμοποιήσουμε έναν μονοδιάστατο πίνακα 81 θέσεων για τους πιθανούς συνδυασμούς, ενώ για τις επιλεγμένες κάρτες θα χρησιμοποιήσουμε έναν δισδιάστατο πίνακα, όπως θα δούμε παρακάτω, όπου κάθε στοιχείο του πίνακα θα αντιστοιχεί σε έναν αριθμό 0 έως 80 ο οποίος θα αντιστοιχεί στο συγκεκριμένο συνδυασμό του πίνακα τύπου **CARD**.



Εδώ φτιάχνουμε τους συνδυασμούς των καρτών. Χρησιμοποιούμε ένα **while** το οποίο θα τρέξει 81 φορές (0 έως 80) και μέσα σε αυτό υπάρχουν 4 nested **for**. Στο τέταρτο **for** έχουμε πλέον έναν καινούργιο συνδυασμό, τον οποίο τον αποθηκεύουμε στην αντίστοιχη θέση του πίνακα **CARDS**.



Τη συνάρτηση αυτή τη χρησιμοποιούμε κατά τη δημιουργία μιας νέας 12άδας καρτών, για να βεβαιωθούμε πως τη συγκεκριμένη κάρτα δεν την έχουμε χρησιμοποιήσει ήδη. Η συνάρτηση διατρέχει όλο τον πίνακα των επιλεγμένων καρτών και αν βρει ταύτιση επιστρέφει 1, αλλιώς 0.



Εδώ επιλέγουμε με τη χρήση της **rand()** τις 12 κάρτες που θα εμφανιστούν. Για κάθε θέση χρησιμοποιούμε ένα **do..while** το οποίο ελέγχει αν η κάρτα ήδη έχει επιλεχθεί με τη χρήση της **isSelected** που είδαμε παραπάνω. Στο τέλος της συνάρτησης, ο πίνακς **cardBoard[3][4]** θα έχει όλες τις επιλεχθέντες κάρτες.



Εδώ εμφανίζουμε τις κάρτες, κάνοντας χρήση της τιμής που έχει η συγκεκριμένη θέση του **cardsSelected**, όπου όπως είπαμε αντιστοιχεί σε μία τιμή 0 έως 80 του **cardboard**. Έτσι μπορούμε να πάρουμε από το **cardboard** τα χαρακτηριστικά της κάρτας και να τα εμφανίσουμε.



Η συνάρτηση αυτή αρχικοποιεί τον πίνακα **players** με τα στοιχεία των παικτών και μηδενικό σκορ.



Εδώ έχουμε τον έλεγχο αν οι 3 κάρτες που επιλέχθηκαν αποτελούν όντως ένα ΣΕΤ. Για **κάθε** χαρακτηριστικό της κάρτας (shape, color, num, texture) ελέγχουμε αν **δεν** είναι είτε **όλα** ίδια είτε **όλα** διαφορετικά. Αν έστω και **ένα** χαρακτηριστικό δεν είναι ίδιο ή διαφορετικό με τα υπόλοιπα, η συνάρτηση επιστρέφει 0 (λάθος ΣΕΤ), αλλιώς 1 (σωστό ΣΕΤ).



Εδώ περνάμε στον πίνακα **cardsSelected** τις θέσεις των καρτών στον πίνακα **cardsAvailable** από τις κάρτες που έχουν επιλεχθεί, σύμφωνα με τις συντεταγμένες του δισδιάστατου πίνακα και επιστρέφουμε 1 για έγκυρο ΣΕΤ και 0 για λάθος ΣΕΤ χρησιμοποιώντας την **cardSetIsValid** που περιγράψαμε παραπάνω.



Εδώ ελέγχουμε όλο τον πίνακα των παικτών για να δούμε αν κάποιος, και ποιος, παίκτης έφτασε ή ξεπέρασε στη βαθμολογία το 10. Το ‘ποιος’ μας χρησιμεύει όταν εμφανίσουμε το όνομα του νικητή σε περίπτωση που τερματιστεί αυτόματα το παιχνίδι επειδή κάποιος έφτασε στο μέγιστο της βαθμολογίας.



Είμαστε πλέον στη **main()** και αφού ορίσουμε τις απαραίτητες μεταβλητές, αρχικοποιούμε τις 81 κάρτες, τη γεννήτρια τυχαίων αριθμών και τα στοιχεία των παικτών.



Στη συνέχεια, φτιάχνουμε έναν βρόγχο **do..while** ο οποίος εκτελείται όσο ο χρήστης δεν επιλέγει 0 (έξοδος) **και** κανείς παίκτης δεν έχει φτάσει σε σκορ το 10 (**maxScoreReached == -1**). Αν η **init** έχει την τιμή 1, τότε σημαίνει πως είτε τρέχουμε το πρόγραμμα για πρώτη φορά, είτε έχει ζητηθεί από τους παίκτες να δημιουργηθούν νέες κάρτες, οπότε καλείται η **initCardBoard.** Στη συνέχεια εμφανίζουμε τις κάρτες με την **displayCards** και διαβάζουμε την επιλογή του χρήστη. Αν ο χρήστης δεν έχει δώσει 0 (έξοδος) και έχει δώσει έγκυρο **id** παίκτη, ορίζουμε αυτόν τον παίκτη ως **currentPlayerId**.Διαφορετικά, σημαίνει πως έχει επιλέξει να ανακατευθούν πάλι οι κάρτες και να εμφανιστεί νέο σετ, και έτσι βάζουμε την τιμή 1 στην **init** και ξεκινάμε πάλι τον βρόγχο.



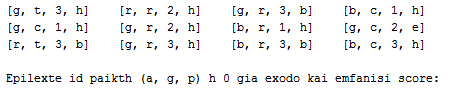
Εδώ είμαστε ακόμα μέσα στον βρόγχο και έχει επιλεχθεί έγκυρος παίκτης. Του ζητάμε να βάλει τις συντεταγμένες των 3 καρτών που θεωρεί ότι αποτελούν ΣΕΤ, ενώ αν οι συντεταγμένες είναι εκτός του αποδεκτού εύρους εμφανίζεται μήνυμα. Με την **checkSet** ελέγχουμε αν το ΣΕΤ είναι σωστό, και αν είναι προσθέτουμε 2 βαθμούς στον παίκτη και βάζουμε στην **init** την τιμή 1 για να εμφανιστεί νέο σετ, αλλιώς αφαιρούμε 1 βαθμό από τον παίκτη μόνο στην περίπτωση που έχει τουλάχιστον έναν βαθμό, και εμφανίζουμε μήνυμα πως θα εμφανιστεί το ίδιο σετ αφού θέτουμε την τιμή 0 στην **init.** Αν οι συντεταγμένες που έχει δώσει ο παίκτης είναι εκτός ορίων εμφανίζεται σχετικό μήνυμα. Μετά έχουμε το τέλος του βρόγχου με τη συνθήκη που περιγράψαμε παραπάνω.



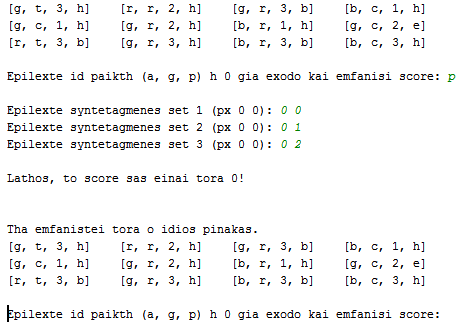
Έξω πλέον από τον βρόγχο, έχει επιλεχθεί είτε η έξοδος, είτε κάποιος παίκτης έφτασε ή ξεπέρασε σε βαθμολογία το 10. Εμφανίζουμε σχετικό μήνυμα, αν επιλέχθηκε η έξοδος, αναλυτικά τα στοιχεία των παικτών και το σκορ τους, και αν κάποιος έχει φτάσει σε βαθμολογία το 10 (**maxScoreReached >= 0**), εμφανίζουμε το όνομά του και το σκορ του.

Ακολουθούν screenshots από την εκτέλεση του προγράμματος.

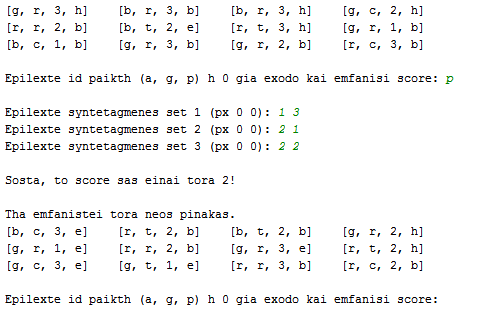
**Screenshot 3.1 – Πρώτη εκτέλεση**



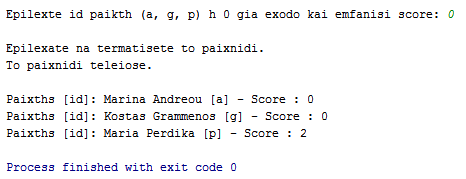
**Screenshot 3.2 – Επιλογή παίκτη με λάθος σετ**



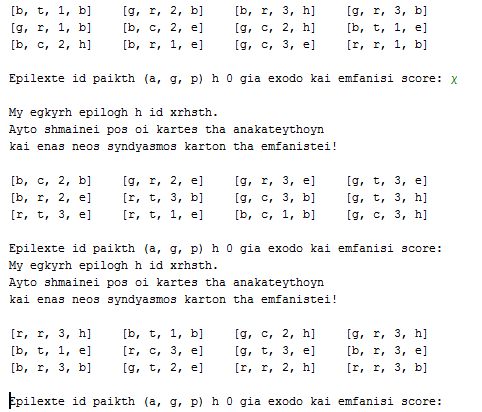
**Screenshot 3.3 – Επιλογή παίκτη με σωστό σετ**



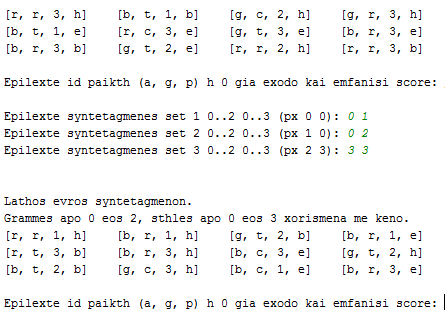
**Screenshot 3.4 – Έξοδος από το παιχνίδι**



**Screenshot 3.5 – Λάθος εισαγωγή και εμφάνιση νέου σετ**



**Screenshot 3.6 – Λάθος συντεταγμένες**



1. https://www.jetbrains.com/clion [↑](#footnote-ref-1)
2. https://www.cygwin.com [↑](#footnote-ref-2)