***ΘΕΜΑΤΙΚΗ ΕΝΟΤΗΤΑ ΠΛΣ50 (2016-17)  
ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ Ε2  
ΤΑΣΙΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ***

**ΕΠΕΞΗΓΗΣΗ ΚΑΙ ΑΝΑΛΥΤΙΚΟΣ ΣΧΟΛΙΑΣΜΟΣ ΘΕΜΑΤΩΝ**

Στο παρακάτω κείμενο επεξηγείται και αναλύεται σχολαστικά ο κώδικας των θεμάτων της Γραπτής Εργασιας Ε2 με αναφορές στις γραμμές του κώδικα.

**ΘΕΜΑ 1: Νέο σύστημα καταγραφής κρατημένων θέσεων**

Στις γραμμές 8 έως και 12 του κώδικα, ορίζεται ένα struct το οποίο θα αποθηκεύει τα απαραίτητα στοιχεία για κάθε θέση του λεωφορείου. Επιπλέον, χρησιμοποιώντας το typedef ορίζουμε αυτό το struct να έχει τον τύπο thesh. Στις γραμμές 14 έως και 21, δηλώνονται οι συναρτήσεις που θα χρησιμοποιηθούν στις διάφορες λειτουργίες του προγράμματος και θα αναλυθούν παρακάτω. Το πρόγραμμα ξεκινά στη γραμμή 23. Στις γραμμές 26 έως και 41 δηλώνονται οι μεταβλητές του προγράμματος και κάποιες από αυτές αρχικοποιούνται. Συγκεκριμένα, στη γραμμή 37, η συνάρτηση readfile αρχικοποιεί τους πίνακες bus[], stoixeia[] (ο πρώτος περιέχει τις θέσεις του λεωφορείου, ο δέυτερος την πινακίδα και τη χωρητικότητα του λεωφορείου) και τη μεταβλητή xwrhtikothta, η οποία είναι ο αριθμός θέσεων του λεωφορείου. Στις γραμμές 42 έως και 113 υπάρχει μία δομή επανάληψης while, η οποία τερματίζεται όταν η μεταβλητή loop γίνει μηδέν (επιλέξει ο χρήστης έξοδο από το πρόγραμμα). Εντός της while, στις γραμμές 44 έως και 54 τυπώνεται το μενού επιλογών του προγράμματος, και διαβάζεται η επιλογή που εισάγει ο χρήστης. Στις γραμμές 55 έως και 112 υπάρχει μια δομή επιλογής switch εντός της οποίας ανάλογα με την εισαγωγή του χρήστη (μεταβλητή choice) γίνονται τα ακόλουθα:  
case 1: Χρησιμοποιείται η συνάρτηση emfanisi\_kenwn με ορίσματα τον πίνακα θέσεων του λεωφορείου και τον αριθμό θέσεων του και τυπώνει τις κενές θέσεις.  
case 2: Ο χρήστης προτρέπεται να δώσει τον αριθμό θέσης που επιθυμεί να κάνει κράτηση. Αν η θέση είναι εκτός ορίων τυπώνεται προειδοποιητικό μήνυμα και προτρέπεται να δώσει ξανά αριθμό θέσης. Αυτό επιτυγχάνεται μέσω μιας δομής επανάληψης do-while. Αν δοθεί αριθμός εντός ορίων, στον δείκτη thesh\* p εκχωρείται η διεύθυνση της θέσης προς κράτηση από τον πίνακα bus. Έπειτα ο δείκτης p χρησιμοποιείται σαν όρισμα στη συνάρτηση eggrafh η οποία εκχωρεί τιμή στη μεταβλητή eggr. Αν η eggr έχει την τιμή 0, ο χρήστης προειδοποιείται ότι η θέση που επιθυμεί είναι ήδη κρατημένη. Αν έχει την τιμή 1, ο χρήστης ενημερώνεται ότι έγινε επιτυχής κράτηση της θέσης που επιθυμεί. Και στις δύο περιπτώσεις, το πρόγραμμα εξέρχεται της switch και επιστρέφει στο μενού επιλογών.  
case 3: Χρησιμοποιείται η συνάρτηση anazhthsh με ορίσματα τον πίνακα θέσεων του λεωφορείου και και τον αριθμό θέσεων του και τυπώνει τα αποτελέσματα της αναζήτησης που έκανε στον πίνακα θέσεων του λεωφορείου.  
case 4: Η case 4 έχει πανομοιότυπη λειτουργία με αυτή της case 2, με τη μόνη διαφορά ότι ο χρήστης δίνει αριθμό θέσης προς ακύρωση.  
case 5: Χρησιμοποείται η συνάρτηση lista\_krathmenwn με ορίσματα τον πίνακα θέσεων του λεωφορείου και τον αριθμό θέσεων του και τυπώνει λίστα με τις κρατημένες θέσεις.  
case 0: Χρησιμοποιείται η συνάρτηση writefile με ορίσματα τον πίνακα θέσεων του λεωφορείου,τον αριθμό θέσεων του και τον πίνακα με τα στοιχεία του λεωφορείου και γράφει στο αρχείο bus.txt. τα στοιχεία αυτά. Έπειτα, η μεταβλητή loop παίρνει τιμή από τη συνάρτηση telos, προκειμένου το πρόγραμμα να εξέλθει από τη while.  
 To πρόγραμμα επιστρέφει την τιμή μηδέν στη γραμμή 115 και τερματίζεται. Όσον αφορά την επεξήγηση των συναρτήσεων:  
**readfile:** Ο τύπος της είναι void γιατί δεν επιστρέφει κάτι στο κύριο πρόγραμμα. Παίρνει σαν ορίσματα του δείκτες του πίνακα των θέσεων του λεωφορείου (γραμμή 34), του πίνακα των στοιχείων του λεωφορείου (γραμμή 35) και της μεταβλητής που κρατάει το πλήθος των θέσεων του λεωφορείου (γραμμή 36). Στις γραμμές 120 έως και 125 ορίζεται ένας δείκτης αρχείου για το bus.txt για ανάγνωση του αρχείου και σε περίπτωση που δεν υπάρχει το αρχείο, εμφανίζεται προειδοποιητικό μήνυμα και η συνάρτηση τερματίζεται. Στις γραμμές 131 έως και 135, το πρόγραμμα διαβάζει το αρχείο χαρακτήρα-χαρακτήρα και εκχωρεί τους χαρακτήρες στον πίνακα stoixeia[] μέχρι να βρεθεί είτε ο χαρακτήρας αλλαγής γραμμής (σε περίπτωση που υπάρχουν στοιχεία κρατήμένων θέσεων στο bus.txt), είτε ο χαρακτήρας EOF (δεν υπάρχουν στοιχεία κρατημένων θέσεων). Έπειτα, στις γραμμές 136 έως και 139, τυπώνονται αυτά τα στοιχεία στην οθόνη. Εφόσον θεωρούμε δεδομένο από την εκφώνηση της άσκησης ότι τα στοχεία για το λεωφορείο είναι έγκυρα, συνεπώς και ο αριθμός θέσεων μικρότερος ή ίσος του 53, η χωρητικότητα του λεωφορείου θα είναι έιτε μονοψήφια έιτε διψήφια. Συνεπώς ο πίνακας stoixeia[] θα έχει μέγιστο μέγεθος 11 ψηφία (προσμετρώ και τον χαρακτήρα ‘\0’ που προσθέτω στο τέλος της συμβολοσειράς στη γραμμή 149 για να μπορώ να χειριστώ τον πίνακα stoixeia[] ως string). Με βάση αυτόν τον συλλογισμό, στις γραμμές 141 έως και 148 υπολογίζω και εκχωρώ τον αριθμό θέσεων στη μεταβλητή xwrhtikothta μέσω του pointer pxwr. Στις γραμμές 151 έως και 155, αρχικοποιώ όλες τις θέσεις του πίνακα bus[], τοποθετώντας σε κάθε ονοματεπώνυμο (member onom\_epwn) την κενή συμβολοσειρά και σε κάθε θέση (member seat) τον αριθμό θέσης ξεκινώντας από το 1 έως και το 53. Για την ανάγνωση των τυχόν γραμμών με τις κρατημένες θέσεις χρησιμοποιείται ο κώδικας στις γραμμές 156 έως και 218: Σύμφωνα με το μέγιστο πλήθος της γραμμής (ονοματεπώνυμο+κενό+αριθμός θέσης+κενό+τηλέφωνο = 53, διότι μέγιστο μήκος ονοματεπώνυμου (επώνυμο+[κενό]+όνομα) 39, μηκος τηλεφώνου 10, μέγιστο μήκος αριθμού θέσης 2) ορίζω εναν πίνακα τύπου char line[] 54 χαρακτήρων ώστε να τοποθετήσω στο τέλος του τον χαρακτήρα ‘\0’ για τον χειρισμό του ως string. Στη γραμμή 157 υπάρχει μία δομή ελέγχου if για να ελέγξει αν το αρχείο δεν έχει φτάσει στο τέλος του μετά την ανάγνωση των στοιχείων του λεωφορείου. Έπειτα, ο πίνακας line[] γεμίζει χαρακτήρα-χαρακτήρα μέχρι να βρεθεί χαρακτήρας αλλαγής γραμμής ή το τέλος του αρχείου. Έπειτα, αν ο πίνακας line έχει χαρακτήρες (δηλαδή δεν υπάρχει απλά στο αρχείο αλλαγή γραμμής και μετά το αρχείο τελειώνει, διότι στο αρχείο δείγμα bus.txt μετά τη γραμμή της τελευταίας κράτησης υπάρχει χαρακτήρας αλλαγής γραμμής.) κάνω διαχωρισμό της γραμμής σε τρεις προσωρινές μεταβλητές για το ονοματεπώνυμο, τον αριθμό θέσης και το τηλέφωνο ανάλογα με τους χαρακτήρες κενού που υπάρχουν στον πίνακα line[] (δηλαδή: επώνυμο[κενό]όνομα[κενό]αριθμός θέσης[κενό]τηλέφωνο). Τοποθέτησα πρώτα τα στοιχεία σε προσωρινές μεταβλητές, επειδή αν δεν έχω εξάγει το αριθμό θέσης δεν μπορώ να ξέρω εκ των προτέρων πού να τοποθετήσω, πχ το ονοματεπώνυμο. Έτσι, για ομοιογένεια στον κώδικα, πρώτα εξάγω και τα τρία δεδομένα που χρειάζομαι σε προσωρινές μεταβλητές και έπειτα τα τοποθετώ στα αντίστοιχα members. Στη γραμμή 219 κλείνει το αρχείο.  
*Σημείωση 1:* Θεωρώ δεδομένο ότι το τηλέφωνο θα έχει πάντα μήκος 10 ψηφίων  
*Σημείωση 2:*Παρά το ότι στο αρχείο-δείγμα bus.txt υπάρχει αλλαγή γραμμής μετά την τελευταία κράτηση, το πρόγραμμά μου χειρίζεται και αρχεία χωρίς αλλαγή γραμμής στο τέλος του.  
**emfanisi\_kenwn:** Η συνάρτηση αυτή παίρνει ως ορίσματα τον πίνακα θέσεων του λεωφορείου και τον αριθμό θέσεων του και τυπώνει όσες θέσεις έχουν για ονοματεπώνυμο την κενή συμβολοσειρά (έλεγχος με τη συνάρτηση strcmp), μετρώντας παράλληλα πόσες είναι οι κενές και τυπώνει το πλήθος τους στο τέλος. Εφόσον ο πίνακας διατρέχεται από την αρχή μέχρι το τέλος του, οι θέσεις θα τυπώνονται ταξινομημένες κατά αριθμό θέσης.  
**eggrafh:** Η συνάρτηση αυτή παίρνει ως όρισμα το δείκτη που «δείχνει» τη θέση που επιθυμούμε να κάνουμε κράτηση (όπως αναφέραμε στην case 2). Αρχικά, ελέγχεται στη γραμμή 245 αν το member onom\_epwn έχει την κενή συμβολοσειρά, δηλαδή αν η συγκεκριμένη θέση είναι κενή. Αν η θέση είναι κενή, ο χρήστης προτρέπεται να δώσει το ονοματεπώνυμο που θέλει να κάνει την κράτηση. Αφότου γίνει χρήση της συνάρτησης fflush στο stream της εισόδου, χρησιμοποιείται η συνάρτηση fgets για αποθήκευση του ονοματεπώνυμου, που εισάγεται από τον χρήστη, σε έναν προσωρινό char πίνακα buffer[]. Επειδή η fgets αποθηκεύει στο buffer και τον χαρακτήρα αλλαγής γραμμής, χρησιμοποίησα την εξής τεχνική ώστε να μην αποθηκευτεί σαν χαρακτήρας του ονοματεπώνυμου: Αρχικά θεωρούμε δεδομένο ότι το μέγιστο μήκος του ονοματεπώνυμου θα είναι 39 χαρακτήρες (επώνυμο+[κενό]+όνομα), συνεπώς δεν υπάρχει έλεγχος αν ο χρήστης δώσει μεγαλύτερη είσοδο σαν ονοματεπώνυμο. Επειδή η fgets αποθηκεύει στο buffer (μήκους 40 χαρακτήρων) 39 χαρακτήρες και στο τέλος τον χαρακτήρα ‘\0’, ελέγχουμε αρχικά αν ο τελευταίος χαρακτήρας πριν τον ‘\0’ είναι ο ‘\n’, (ισχύει για εισαγωγή του χρήστη μικρότερη των 39 χαρακτήρων). Αν ισχύει αυτό, αποθηκεύεται στο member onom\_epwn το περιεχόμενο του buffer μέχρι τον χαρακτήρα ‘\n’ και έπειτα ο ειδικός χαρακτήρας ‘\0’. Αν η είσοδος είναι 39 χαρακτήρες (οπότε δεν αποθηκέυεται ο χαρακτήρας ‘\n’ στο buffer), το περιεχόμενο του buffer αποθηκεύεται όπως είναι στο onom\_epwn. (Επειδή θεωρούμε έγκυρη την είσοδο του ονοματεπώνυμου χρησιμοποιούμε τη strcpy, χωρίς να φοβόμαστε ότι θα υπάρξει κάποιο πρόβλημα στην αποθήκευσή του στο struct). Στη γραμμή 260 ο χρήστης προτρέπεται να δώσει τον αριθμό τηλεφώνου. Αφού καθαρίστεί το stream εισόδου με την fflush, η fgets διαβάζει το τηλέφωνο, το οποίο θεωρούμε δεδομένο ότι θα έχει πάντα μήκος 10 ψηφίων και ότι η εισαγωγή του θα είναι έγκυρη (συνεπώς δεν υπάρχει περαιτέρω έλεγχος). Αφού διαβαστεί το τηλέφωνο, τα ψηφία αποθηκεύονται με τη σειρά στο member phone[].Τέλος, επιστρέφεται η τιμή 1 για επιτυχημένη κράτηση ή η τιμή μηδέν για αδυναμία κράτησης.  
**anazhthsh:** Στη γραμμή 284 ο χρήστης προτρέπεται να επιλέξει κριτήριο αναζήτησης και πραγματοποιείται έλεγχος ώστε να εισάγει έγκυρη επιλογή (το 1 ή το 2). Μετά την επιλογή του κριτηρίου, η εισαγωγή του προς αναζήτηση ονοματεπώνυμου ή τηλεφώνου γίνεται με παρόμοιο τρόπο με την εισαγωγή αυτών στη συνάρτηση eggrafh, με τη μόνη διαφορά ότι αποθηκεύονται σε προσωρινές μεταβλητές firlasname[] και thlefwno[] για την αναζήτησή τους πίνακα bus[]. Ο έλεγχος για το ονοματεπώνυμο γίνεται μέσω της μεταβλητής strcmp, ενώ όσον αφορά το τηλέφωνο γίνεται έλεχος ψηφίο-ψηφίο και μέσω της μεταβλητής-σημαίας thlfound διαπιστώνεται αν υπάρχει κράτηση σε συγκεκριμένο τηλέφωνο.  
**diagrafh:** Η συνάρτηση αυτή παίρνει σαν όρισμα τον δείκτη της θέσης που επιθυμούμε να ακυρώσουμε. Αρχικά, γίνεται έλεγχος αν η θέση είναι όντως κρατημένη. Αν είναι κρατημένη, στο onom\_epwn εισάγεται η κενή συμβολοσειρά, στο phone[] εισάγονται μηδενικά και επιστρέφεται η τιμή 1. Αν δεν είναι κρατημένη, επιστρέφεται η τιμή μηδέν*. Σημείωση:* Ενώ κατά την αρχικοποίηση των θέσεων του λεωφορείου δεν εισήγαγα συγκεκριμένες τιμές στο member phone[] των ελεύθερων θέσεων και άφησα όποιες τιμές προϋπήρχαν, κατά τη διαγραφή συγκεκριμένης θέσης γέμισα το phone[] με μηδενικά, διότι το τηλέφωνο αποτελεί κριτήριο αναζήτησης και αν συνεχίζει να υφίσταται το τηλέφωνο της διεγραμμένης θέσης, θα φαίνεται λανθασμένα ότι υπάρχει ακόμα κράτηση στο συγκεκριμένο τηλέφωνο.  
**lista\_krathmenwn** Αρχικά, γίνεται έλεγχος αν υπάρχουν κρατημένες θέσεις. Αν δεν υπάρχουν κρατημένες θέσεις τυπώνεται το ανάλογο μήνυμα. Αν υπάρχουν, το τηλέφωνο κάθε κρατημένης θέσης αποθηκεύεται σε μια συμβολοσειρά κάθε φορά και οι κρατήσεις τυπώνονται ταξινομημένες κατά αριθμό θέσης (αφού ο πίνακας διατρέχεται από την αρχή μέχρι το τέλος του).  
**writefile:** Η συνάρτηση παίρνει ως ορίσματα τον πίνακα θέσεων του λεωφορείου, τον πίνακα με τα στοιχεία του λεωφορείου και τη μεταβλητή της χωρητικότητάς του. Αρχικά ορίζεται ο δείκτης αρχείου arxeio για την εγγραφή στο bus.txt. Έπειτα, εγγράφονται στο αρχείο τα στοιχεία του λεωφορείου και οι κρατημένες θέσεις στη μορφή του δείγματος bus.txt. Επειδή στο αρχείο-δείγμα ο τελευταίος χαρακτήρας είναι η αλλαγή γραμμής, το παραγόμενο αρχείο τελειώνει και αυτό με αλλαγή γραμμής.  
**telos:** Η συνάρτηση αυτή επιστρέφει την τιμή μηδέν, η οποία εκχωρείται στη μεταβλητή loop στο κύριο πρόγραμμα, προκειμένου το πρόγραμμα να εξέλθει της while και να τερματιστεί.

**ΘΕΜΑ 2: Επέκταση συστήματος καταγραφής κρατημένων θέσεων**

Το Θέμα 2 μιας και αποτελεί επέκταση του Θέματος 1, έχει κώδικα παρόμοιο με αυτόν του Θέματος 1. Οι διαφορές παρατηρούνται στα εξής:

-Στις γραμμές 15 έως και 18 προστέθηκε το struct με τον τύπο list για την κατασκευή της μονά συνδεδεμένης λίστας.  
- Στη γραμμή 20, η συναρτηση readfile έχει τύπο thesh\*, διότι επιστρέφει ένα δείκτη τύπου thehs. (Επεξήγηση της νέας συνάρτησης παράκάτω)  
- Ο πίνακας των στοιχείων του λεωφορείου stoixeia[] έχει χωρητικότητα 13 χαρακτήρων. Σ’ αυτό το σημείο θα ήθελα να αναφέρω ότι από τα δεδομένα του προβλήματος, επειδή η νέα χωρητικότητα του λεωφορείου δεν έχεi περιορισμό και επειδή η μεταβλητή της θέσης του επιβάτη είναι τύπου unsigned int, η τελευταία πιθανή θέση του λεωφορείου θα είναι η 232-1. Επειδή, όμως στο δικό μου υπολογιστή η συγκεκριμένη τιμή προκαλεί πρόβλημα στο compilation του προγράμματος και επειδή το πρόγραμμα γίνεται αισθητά πιο αργό με την αύξηση του αριθμού θέσεων (χρησιμοποιούνται αλγόριθμοι γραμμικής αναζήτησης και ταξινόμησης φυσσαλίδας), θεώρησα (χωρίς να πιστέυω ότι προκαλείται βλάβη της γενικότητας του προβλήματος) ότι η μέγιστη τιμή που μπορεί να πάρει η χωρητικότητα του λεωφορείου είναι 9999 θέσεις.  
- Για την προσπέλαση του πίνακα θέσεων του λεωφορείου χρησιμοποιείται ο δείκτης p1 που ορίζεται στη γραμμή 35 και αρχικοποιείται μέσω της συνάρτησης readfile στη γραμμή 43. Επίσης, στις συναρτήσεις που έχουν την ίδια λειτουργία και στα δύο θέματα, εκεί που στο Θέμα 1 έπαιρναν ως όρισμα τον πίνακα θέσεων του λεωφορείου, τώρα παίρνουν ως όρισμα τον δείκτη p1 του πίνακα θέσεων.  
- Στην case 5, στις γραμμές 113 έως και 117, υπάρχει μία δομή do-while εντός της οποίας ο χρήστης εισάγει επιλογή για ταξινόμηση κατά ονοματεπώνυμο ή κατά αριθμό θέσης. Το πρόγραμμα εξέρχεται από τη do-while μόνο αν ο χρήστης δώσει έγκυρη επιλογή (είτε το μηδέν είτε το 1). Έπειτα ακολουθεί ταξινόμηση μέσω της συνάρτησης bubblesort και τύπωση της λίστας μέσω της συνάρτησης lista\_krathmenwn. Αν έχει προηγηθεί ταξινόμηση κατά ονοματεπώνυμο (η μεταβλητή tax, που έχει αποθηκευμένη την επιλογή του χρήστη, έχει την τιμή μηδέν) χρησιμοποιείται ξανά η bubblesort για ταξινόμηση κατά αριθμό θέσης όπως επιβάλλουν οι απαιτήσεις του προγράμμματος.  
- Στην case 0, ο πίνακας ταξινομείται αρχικά κατά ονοματεπώνυμο και έπειτα δημιουργείται η μονά συνδεδεμένη λίστα μέσω της συνάρτησης add\_to\_list με κεφαλή τη μεταβλητή head.  
- Όσον αφορά τις αλλαγές στις συναρτήσεις:  
**readfile:** Οι αλλαγές παρατηρούνται στα εξής: στις γραμμες 161 έως και 175 επειδή όπως προανέφερα υπάρχει πρόβλεψη για 4ψήφια χωρητικότητα, υπολογίζονται όλες οι περιπτώσεις χωρητικότητας μέχρι 9999 θέσεις. Επίσης στη γραμμή 178 ορίζεται ο δείκτης του δυναμικού πίνακα, τύπου thesh\*, pbus στον οποίο εκχωρούνται τόσες θέσεις μνήμης, όσες και οι θέσεις του λεωφορείου. Στις γραμμές 179 εώς και 183, ο πίνακας αρχικοποιείται με όλες τις θέσεις κενές. Η συνάρτηση αυτή επιστρέφει τον δείκτη pbus ο οποίος δείχνει στον πίνακα θέσεων του λεωφορείου.  
-**bubblesort:** Η συνάρτηση bubblesort (υπάρχει μόνο στο Θέμα 2) είναι ο κλασικός αλγόριθμος ταξινόμησης φυσσαλίδας που με τη χρήση δύο δεικτών σαρώνει τον πίνακα και φέρνει στην «επιφάνεια» τα μικρότερα στοιχεία. Το διαφορετικό σε αυτή τη συνάρτηση έγκειται στο γεγονός ότι ως όρισμα παίρνει και την επιλογή του χρήστη στην case 5 (ταξινόμηση κατά ονοματεπώνυμο ή αριθμό θέσης). Έτσι ανάλογα με την επιλογή του χρήστη πραγματοποιείται η ανάλογη ταξινόμηση.  
**add\_to\_list:** Η συνάρτηση αυτή (υπάρχει μόνο στο Θέμα 2) παίρνει ως ορίσματα την κεφαλή (μεταβλητή head) που ορίσαμε στο κύριο πρόγραμμα, τον δείκτη του πίνακα θέσεων του λεωφορείου, τη χωρητικότητα του λεωφορείου και μια μεταβλητή i, η οποία θα μας βοηθήσει στην καταχώρηση των κρατημένων θέσεων στη λίστα. Η base case είναι όταν η μεταβλητή i φτάσει τη χωρητικότητα του λεωφορείου, οπότε και σταματάει η αναδρομικότητα της συνάρτησης και επιστρέφεται η μεταβλητή root, δηλαδή η μεταβλητή head του κύριου προγράμματος στην οποία έχουν συνδεθεί οι κόμβοι της λίστας. Αν το i δεν έχει φτάσει τη χωρητικότητα, όσο δεν υπάρχουν κρατήσεις, το i θα αυξάνεται κατά ένα (γραμμες 468 έως και 471). Μόλις βρεθεί κρατημένη θέση, στη γραμμή 472 εκχωρείται μία θέση μνήμης για τον κόμβο και στη γραμμή 473 εκχωρείται το περιεχόμενο της θέσης του λεωφορείου στο member epivaths της μεταβλητής root. Έπειτα στη γραμμή 475, στο member next της root εκχωρείται η συνάρτηση add\_to\_list με ορίσματα τον δείκτη του πίνακα θέσεων του λεωφορείου, τη μεταβλητή i αυξημένη κατά 1 και το member next της root, υλοποιώντας έτσι το αναδρομικό κομμάτι της συνάρτησης. Η μεταβλητή i θα μπορούσε να μην αρχικοποιηθεί στο μηδέν, αλλα να αρχικοποιηθεί στην τιμή: ([χωρητικότητα λεωφορείου] – [αριθμός κρατημένων θέσεων]), διότι με αύξουσα ταξινόμηση κατά ονοματεπώνυμο, οι θέσεις με κενή συμβολοσειρά “\n” θα βρισκόντουσαν στις πρώτες θέσεις του πίνακα.  
**writefile:** Η συνάρτηση αυτή για την εγγραφή στο αρχείο χρησιμοποιεί την απλά συνδεδεμένη λίστα. Η δομή while, που ξεκινάει στη γραμμή 492, έχει ως συνθήκη η μεταβλητή root να μην είναι NULL, δηλαδή η λίστα να μη φτάσει στο τέλος της. Στις γραμμές 498 έως και 511, ο αριθμός θέσης του επιβάτη μετατρέπεται από unsigned int σε ψηφία (τύπου char) προκειμένου να εγγραφεί στο αρχείο. Υπενθυμίζεται ότι το πρόγραμμα χειρίζεται αριθμούς θέσεις μέχρι το 9999. Έτσι, ο αριθμός θέσης με διαδοχικές διαιρέσεις με δυνάμεις του 10 μετατρέπεται σε ψηφία. Στη γραμμή 522, στη μεταβλητή root εκχωρείται το member next , έτσι ώστε να δείχνει στον επόμενο κόμβο της λίστας.

**ΘΕΜΑ 3: Υλοποίηση Παραλλαγής Παιχνιδιού SET**

Στις γραμμές έως 10 έως και 14 ορίζεται το struct με τον τύπο paixths για τον ορισμό των ιδιοτήτων κάθε παίχτη.Σημειώνεται ότι επειδή η βαθμολογία κάθε παίχτη κυμαίνεται μεταξύ 0 και 255, όρισα το member score με τύπο unsigned char, αφού οι τιμές που μπορεί να πάρει μια μεταβλητή τέτοιου τύπου είναι μεταξύ 0 και 255. Στις γραμμές 16 έως και 21 ορίζεται το struct με τον τύπο karta για τον ορισμό των ιδιοτήτων κάθε κάρτας. Στις γραμμές 24 έως και 26 βρίσκονται οι ορισμοί τριών συναρτήσεων: της getNextcard η οποία επιλέγει με τυχαίο τρόπο μία κάρτα, της gameTable που δημιουργεί και τυπώνει το πλέγμα και της elegxosSet η οποία ελέγχει αν 3 κάρτες είναι σετ (ανάλυση των συναρτήσεων παρακάτω). Το πρόγραμμα ξεκινά στη γραμμή 29. Στις γραμμές 31 έως και 40 ορίζεται και αρχικοποιείται ο πίνακας players[] που περιέχει τα στοιχεία κάθε παίχτη. Στις γραμμές 41 έως και 44 ορίζονται οι πίνακες xrwmata[], sxhmata[], arithmoi[] και yfes[] που περιέχουν τις πιθανές τιμές για κάθε ιδιότητα κάποιας κάρτας. Στις γραμμές 52 έως και 68 υπάρχουν τέσσερις δομές επανάληψης for, εμφωλευμένες η μία στην άλλη, εντός των οποίων αρχικοποιείται ο πίνακας kartes[] ο οποίος περιέχει του 81 πιθανούς συνδυασμούς καρτών που υπάρχουν στο παιχνίδι. Στη γραμμή 69, γίνεται χρήση της srand με όρισμα το time(NULL) προκειμένου να τροφοδοτηθεί η συνάρτηση rand που θα χρησιμοποιηθεί αργότερα στο πρόγραμμα. Στη γραμμή 71 ορίζεται ο δισδιάστατος πίνακας plegma που θα αποθηκέυει την τυχαία 12άδα που θα επιλέγει το πρόγραμμα. Στη γραμμή 74 χρησιμοποιείται η gameTable για την παραγωγή της πρώτης 12άδας του παιχνιδιού. Ο κύριος κώδικας του παιχνιδιού βρίσκεται εντός της δομής while, στις γραμμές 76 έως και 146, με συνθήκη να μην έχει φτάσει η βαθμολογία κάποιου παίχτη (δηλαδή το member score) την τιμή 10. Στις γραμμές 78 έως και 81, ο χρήστης προτρέπεται να δώσει το id του και το πρόγραμμα το διαβάζει. Επειδή οι απαιτήσεις δεν αναφέρουν να γίνει έλεγχος της εισαγωγής του id του χρήστη ούτε ως προς το είδος της εισαγωγής, ούτε αν το id που εισάγεται υπάρχει στους καταγεγραμμένους χρήστες, θεωρώ ότι το id είναι έγκυρο. Στις γραμμές 82 έως και 85, αν ο χρήστης έχει εισάγει το μηδέν τότε το πρόγραμμα εξέρχεται από τη while (το παιχνίδι τελειώνει). Στις γραμμές 87 έως και 91, αν ο παίχτης εισάγει ‘!’, τυπώνεται μία νέα 12άδα και με την εντολή continue η ροή του προγράμματος μεταφέρεται στην αρχή της δομής while. Στις γραμμές 93 έως και 100, γίνεται μία γραμμική αναζήτηση του id που έχει εισαχθεί, ώστε να αντιστοιχιστεί με τον παίχτη του και αυτός να οριστεί ως ενεργός ( ώστε να μπορώ να αλλάξω και τη βαθμολογία του αργότερα). Στις γραμμές 102 έως και 124 υπάρχουν τρεις δομές επανάληψης do-while, εντός των οποίων ζητείται από τον ενεργό παίχτη να δώσει τις συντεταγμένες των καρτών, των οποίων θεωρεί σετ, στη μορφή: γραμμή στήλη. Μέσω των δομών επανάληψης γίνεται έλεγχος αν οι συνεταγμένες που δόθηκαν είναι εκτός του πλέγματος, οπότε και ζητούνται ξανά οι συντεταγμένες. Οι συντεταγμένες αποθηκεύονται στον πίνακα synt[] (που ορίστηκε στη γραμμή 50). Στις γραμμές 126 έως και 128, με δεδομένο τις συντεταγμένες των τριών καρτών, αρχικοποιούνται τρεις μεταβλητές (τύπου karta) p1, p2 και p3 οι οποίες χρησιμοποιούνται ως ορίσματα στη συνάρτηση elegxosSet. Η συνάρτηση elegxosSet εκχωρεί τιμή στη μεταβλητή points. Αν η τιμή που της εκχωρήθηκε είναι το 2, τυπώνεται μήνυμα ότι βρέθηκε σετ, η βαθμολογία του παίχτη αυξάνεται κατά 2 και τυπώνεται νέο πλέγμα μέσω της συνάρτησης gameTable. Σε διαφορετική περίπτωση, τυπώνεται μήνυμα ότι δε βρέθηκε σετ και αν η βαθμολογία του ενεργού παίχτη είναι μεγαλύτερη του μηδενός, τότε μειώνεται κατά 1 (αυτό δε φαίνεται άμεσα στη γραμμή 143, αλλά η συνάρτηση elegxosSet επιστρέφει είτε την τιμή 2, είτε την τιμή -1). Στις γραμμές 149 έως και 155, ο πίνακας των παιχτών διατρέχεται ώστε να διαπιστωθεί αν υπάρχει νικητής και να τυπωθεί το μήνυμα: «O nikhths einai o [ονομα πάιχτη]!». Στις γραμμές 157 έως και 162 τυπώνονται οι βαθμολογίες των παιχτών μέσω μιας δομής επανάληψης for. Όσον αφορά τις συναρτήσεις:  
**getNextCard:** Παίρνει ως ορίσματα τον δείκτη του πίνακα με τις 81 κάρτες και τον δείκτη, μιας μεταβλητής τύπου κάρτα, ptemp (που ορίζεται εντός της συνάρτησης gameTable). Η μεταβλητή a παίρνει μια τυχαία τιμή (έστω ν) από 0 έως και 80 και έπειτα στη μεταβλητή που δείχνει ο ptemp εκχωρείται η ν-οστη κάρτα του πίνακα καρτών. Η συνάρτηση επιστρέφει την τιμη a.  
**gameTable:** Παίρνει ως ορίσματα τον δείκτη του πλέγματος των καρτών και τον δείκτη του πίνακα των καρτών. Στη γραμμή 178, ορίζεται ο πίνακας deck[] στον οποίο θα αποθηκεύονται οι θέσεις που έχουν οι κάρτες στον πίνακα kartes[] (θα χρησιμεύσει στο να αποθηκεύονται μοναδικές κάρτες στο πλέγμα). Στις γραμμές 181 έως και 197, για κάθε θέση του πλέγματος των 12 καρτών, καλείται η συνάρτηση getNextCard η οποία όπως προαναφέρθηκε αποθηκεύει στη μεταβλητή temp την κάρτα που επιλέχθηκε τυχαία και επιστρέφει τη θέση της. Κατόπιν, γίνεται έλεγχος στον πίνακα deck[] αν η συγκεκριμένη κάρτα υπάρχει στο πλέγμα (δηλαδή αν υπάρχει η θέση της στον deck[]). Αν υπάρχει ήδη, καλείται ξανά η getNextCard. Αν δεν υπάρχει, η κάρτα αποθηκεύεται στο πλέγμα και η θέση της στον deck[]. Η διαδικασία συνεχίζεται μέχρι να αποθηκευτούν στο πλέγμα 12 μοναδικές κάρτες. Στις γραμμές 198 έως και 208 τυπώνεται το πλέγμα στην οθόνη.  
**elegxosSet:** Παίρνει ως ορίσματα τρεις μεταβλητές τύπου karta οι οποίες έχουν αποθηκευμένες τις τρεις κάρτες με τις δεδομένες συντεταγμένες που έδωσε ο χρήστης. Έπειτα, γίνονται διαδοχικοί έλεγχοι των ιδιοτήτων των τριών καρτών. Αν συντρέχουν οι προϋποθέσεις ώστε οι κάρτες να είναι ΣΕΤ, επιστρέφεται η τιμή 2. Αλλιώς επιστρέφεται η τιμή -1.

*Γενική Σημείωση:* Σε κάποιες συναρτήσεις των παραπάνω ασκήσεων τα ονόματα των ορισμάτων που χρησιμοποιούνται στην υλοποίηση της συνάρτησης είναι ίδια με τα ονόματα των ορισμάτων που χρησιμοποιούνται στην εφαρμογή της συνάρτησης εντός του κύριου προγράμματος. Κατά κανόνα, γνωρίζω ότι αυτό αποφεύγεται. Εν τούτοις, για να αποφύγω λάθη εκ παραδρομής λόγω πολλών μεταβλητών και συναρτήσεων και επειδή το scope κάθε συνάρτησης και κύριου προγράμματος είναι διαφορετικό, υπέπεσα σε αυτή την «παράλειψη».