Στη δεύτερη άσκηση κινήθηκα ως εξής:

Αρχικά έκανα αντιγραφή τον κώδικα από την πρώτη άσκηση και διέγραψα όλα τα σχόλια!

Αυτό έγινε για να επικεντρωθούμε στις όποιες αλλαγές έχουν προκύψει με νέο σχολιασμό.

Αρχικά «έσπασα» τη δήλωση του τύπου της δομής σε δύο βήματα, καθώς μέσα στη δήλωση, περιέλαβα και έναν δείκτη σε ίδιου τύπου struct και η c δεν γνωρίζει προς στιγμήν τον τύπο του δείκτη. Έτσι προηγήθηκε μια δήλωση του bus\_t με χρήση της typedef και στο δεύτερο βήμα παραλείφθηκε το όνομα bus\_t, αφού αυτό είχε δηλωθεί στο πρώτο βήμα!

Μετά έκανα μια δήλωση global μεταβλητής δείκτη fin σε αρχείο, για να τη «βλέπουν» όλες οι σχετικές συναρτήσεις.

Μετά έγιναν οι πρόσθιες δηλώσεις των συναρτήσεων,που πλέον είναι περισσότερες.Σαν σχόλιο άφησα μια μικρή βοηθητική συνάρτηση print\_lista, προς δική μου διευκόλυνση, με την οποία ελέγχω αν η απλά συνδεδεμένη λίστα μου λειτουργεί σωστά.

Μπαίνοντας στη main δήλωσα μεταξύ άλλων δύο δείκτες σε δομή bus\_t, τον bus, καθώς και τον δείκτη head(που δείχνει την κεφαλή της λίστας)

Μετά καλώντας τη σχετική συνάρτηση άνοιξα το αρχείο και διάβασα πινακίδα και θέσεις. Να σημειώσω πως το αρχείο παραμένει ανοικτό σε αυτό το σημείο.

Με βάση τον αριθμό των θέσεων που μόλις διάβασα από το αρχείο μου, κάνω δυναμική δέσμευση μνήμης με χρήση της malloc.

Στη συνέχεια καλώ τη συνάρτηση αρχικοποίησης του struct και ακολούθως καλώ τη συνάρτηση κράτημα θέσεων από αρχείο και κλείνω τελικά το αρχείο. Η σειρά των κινήσεων με την ενδιάμεση παρεμβολή της malloc, με ώθησαν στο «σπάσιμο» της συνάρτησης ανοίγματος του αρχείου που είχα χρησημοποιήσει στην προηγούμενη άσκηση.

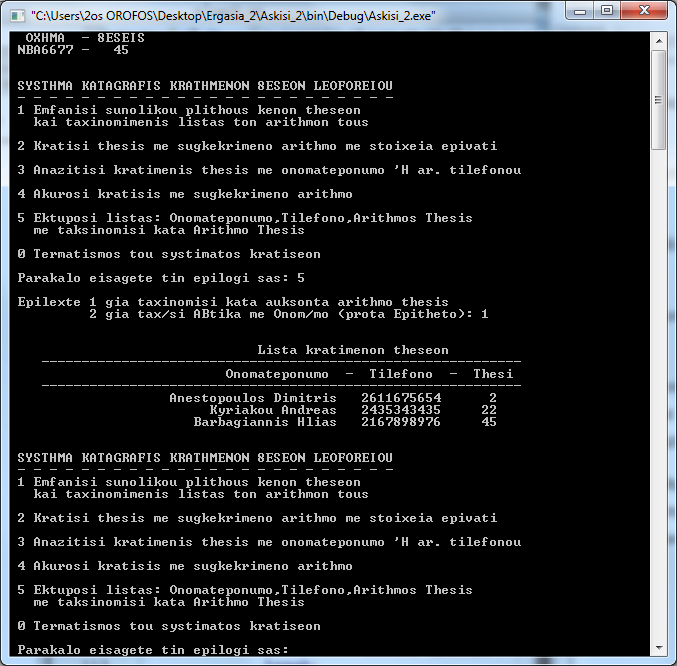
Μετά μπαίνουμε στις επιλογές του προγράμματος και η διαφοροποίηση αφορά την επιλογή 5.Με εσωτερικό switch «σπάμε» σε 2 υποεπιλογές-ταξινόμηση με αρ.θέσης ή με ονοματεπώνυμο. Στην πρώτη περίπτωση απλά καλούμε τη συνάρτηση εκτύπωσης λίστας, καθώς η λίστα μας είναι ήδη ταξινομημένη κατά αρ. θέσης και δε χρειάζεται κάποια αλλαγή. Αντίθετα στη δεύτερη περίπτωση καλώ τη συνάρτηση bubblesort\_eponumo, για να ταξινομήσω με το ονοματεπώνυμο,μετά καλώ την εκτύπωση και τέλος την bubblesort\_thesi για να επαναφέρω την ταξινόμηση κατά αρ. θέσης. Τέλος αν ο χρήστης πατήσει μη αποδεκτό χαρακτήρα, τυπώνω το σχετικό μήνυμα και επανέρχομαι στο loop των επιλογών.

Η νέα συνάρτηση bubblesort\_eponumo λειτουργεί ως εξής:

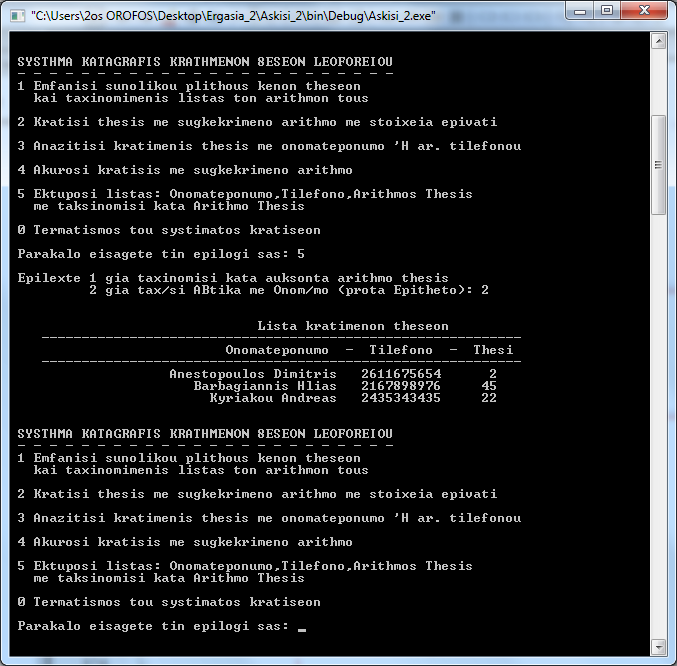
Έχει σαν όρισμα έναν δείκτη στον πίνακα bus (τυπου bus\_t) και το μήκος length του πίνακα. Η συνάρτηση κάνει ταξινόμηση με βάση το επώνυμο. για να το πετύχει αυτό χρησιμοποιεί δύο εμφωλευμένες for. Η εσωτερική for διατρέχει τον πίνακα από το τέλος του, μέχρι μια θέση k, και φέρνει στη θέση k του πίνακα, το μικρότερο αλφαβητικά στοιχείο. Για να το πετύχει αυτό, ξεκινάει από το τέλος του πίνακα και ελέγχει την αλφαβητική σειρά (με τη χρήση της συνάρτησης strcmp())  του πεδίου name του τελευταίου και του προτελευταίου struct του πίνακα. Αν τα δύο struct είναι σε σωστή σειρά, τα αφήνει ως έχουν, αλλιώς τους αλλάζει θέση. Να σημειωθεί ότι η αλλαγή θέσης γίνεται για όλα τα πεδία του struct, με τη χρήση ενδιάμεσων μεταβλητών. Η εξωτερική for αρχικά θέτει k=0 ώστε η εσωτερική for να διατρέξει όλον τον πίνακα. Όταν γίνει αυτό, ξέρουμε ότι στην θέση 0 του πίνακα, βρίσκεται το μικρότερο στοιχείο όλου του πίνακα. Στη συνέχεια το k αυξάνεται κατά ένα και η διαδικασία επαναλαμβάνεται. Η μέσα for σαρώνει όλον τον πίνακα, εκτός από τη θέση 0 και φέρνει στη θέση 1 το δεύτερο μικρότερο στοιχείο. Eπαναλαμβάνουμε συνεχώς για όσο k<length, και τελικά καταλλήγουμε με όλο τον πίνακα ταξινομημένο..

Η συνάρτηση bubblesort\_thesi έχει παρόμοια λογική με μόνη διαφορά κατά τη συνθήκη σύγκρισης ελέγχει τα πεδία b.th της θέσης i και της θέσης (i-1) και αναλόγως κάνει τα «τουμπαρίσματα».

Περίπτωση ταξινόμησης με αρ.θέσης



Περίπτωση ταξινόμησης με ονοματεπώνυμο:



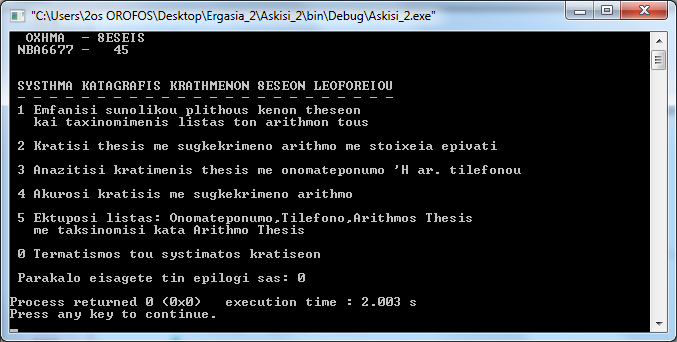
Στη συνέχεια βγαίνοντας από το loop κάνουμε εκ νέου bubblesort\_eponumo και καλούμε και τη νέα συνάρτηση dimiourgia\_listas, με ορίσματα το struct bus το δεικτη head και τις theseis.

Για να δουλέψει η νέα μας συνάρτηση πρέπει πρώτα να κάνουμε εισαγωγή κάθε νέου στοιχείου σε λίστα. Αυτό το πετυχαίνουμε με μια ακόμη συνάρτηση την eisagogi\_se\_lista με ορίσματα 2 δείκτες με αναφορά, ο δείκτης head και o n. Επειδή ο head δείχνει στο NULL και δεν υφίσταται καν σαν κόμβος και επειδή θέλουμε με το πέρας της συνάρτησης να παραμείνει η αλλαγή του πού θα μας δείχνει, χρειαζόμαστε έναν δείκτη του δείκτη head, γι’αυτό και βάζουμε δύο αστέρια!!!. Οπότε πρώτα εξετάζουμε την περίπτωση άδειας λίστας( ο head δείχνει στο NULL) και καταχωρούμε το n struct. Αλλιώς αν ο next του head δείχνει στο NULL(δηλαδή φτάσαμε-μέσω αναδρομικότητας-στο τελευταίο στοιχείο τις λίστας), καταχωρούμε το n στοιχείο στο next του head. Εδώ για ασφάλεια κάνουμε αρχικοποίηση στο next του n να δείχνει στο NULL. Αλλιώς καλούμε εκ νέου την ίδια συνάρτηση με όρισμα εκτός από το n στοιχείο, τη διεύθυνση του next του head! Έτσι μέσω της αναδρομικότητας θα εισάγεται κάθε στοιχείο στο τέλος της λίστας μέχρι ο δείκτης να δείξει στο NULL.

Επιστρέφουμε στη συνάρτηση dimiourgia\_listas, όπου με μια for διατρέχουμε τις θέσεις του λεωφορέιου και όποτε βρίσκουμε κρατημένη θέση(δηλαδή όταν ο πρώτος χαρακτήρας του πεδίου bus[i].name[0] δεν είναι το \0) καλούμε τη συνάρτηση eisagogi\_se\_lista με ορίσματα την κεφαλή της λίστας head και τη διεύθυνση της δομης bus.

Τέλος κατά το κλείσιμο του προγράμματος κάνουμε την αποθήκευση στο αρχείο μέσω της λίστας μας που είναι ταξινομημένη κατά ονοματεπώνυμο. Η διαφορά με την πρώτη άσκηση είναι πως πλέον η συνάρτηση δε διαβάζει από τον πίνακα με τα struct αλλά από τη λίστα μας με τη βοήθεια του δείκτη head. Αντίστοιχα στις αποθηκεύσεις του αρχείου χρησημοποιούμε τα πεδία head->name, head->th και head->num. Όλα μπαίνουν σε ένα loop που τερματίζει όταν head==NULL και σε κάθε γύρο καταχωρεί το head->next στο head.

Κατά την έξοδο από το πρόγραμμα με 0



Η λίστα αποθήκευσε στο αρχείο bus.txt τις κρατημένες θέσεις με ταξινόμηση κατά ονοματεπώνυμο(προσωρινή αποθήκευση-καθώς μετά το επανέφερα στην αρχική!)

