Χρησιμοποιηθηκε το Codeblocks 16.01 σε windows 8.

Παρατηρησεις για θεμα 1 και 2

Στην αρχικοποιηση του πινακα του θεματος 1 επιλεχθηκε στο πεδιο arithmos\_thesis εαν δεν εχει κρατηση να το βαζουμε ισο με -1. Ετσι ολα τα σκαναρισματα του πινακα γιναν αποκλειστηκα και μονο μεσω του δεικτη του μιας και ο arithmos\_thesis εχει τιμη τον αριθμο θεσης μονο εαν ειχε κρατηση. Φυσικα θα μπορουσε απλα να ειναι ισο με τον αριθμο θεση (οπως εκανα στο θεμα 2) και να διευκολυνει στις αναζητησεις/εμφανισεις θεσεων των θεσεων στις επιλογες 2 και 3 ενω στην επιλογη 4 να γλυτωσουμε την αλλαγη του arithmos\_thesis σε -1. Οποτε ισως ηταν ο λιγοτερο αποδοτικος τροπος. Εφοσον βεβαια ακολουθησα την εναλλακτικη μεθοδο στο θεμα 2 τα αφησα και τα 2 προς συλλογη σχολιων απο εσας. Σε καθε περιπτωση οι κωδικες τοσο του θεματος 1 οσο και τους θεματος 2 ειναι λειτουργικοι και το txt καταληγει με τις σωστες καταχωρησεις (οτι τροποποιηση των κρατησεων και αν εκανα μεσω των επιλογων).

# Θεμα 1

Γραμμη 6-10: Η δομη που θα χρησιμοποιηθει για την αναπαρασταση θεσεων με πεδια onomatepwnumo ως character που ειναι array 40 θεσεων για την αποθηκευση του ονοματεπωνυμου, arithmos\_thesis ως unsigned integer για τον αριθμο θεσης και tilefwno ως insigned short που ειναι array 10 θεσεων.

Γραμμη 14-21: function menu() που εμφανιζει με αριθμημενη λιστας τις δυνατες δραστηριοτητες που μπορει να επιλεξει ο χρηστης.

Γραμμη 23-88: function read\_file(..) για αναγνωση και αποθηκευση των στοιχειων του λεοφωρειου απο txt.

Γραμμη 26-28: δηλωση μεταβλητης found ως character που προοριζεται για ελεγχο στο while-loop γραμμη 67-82 οπου σκαναρουμε τον πινακα των θεσεων για το που θα αποθηκευτει η καθε κρατηση. Ακομη δηλωνουμε και τις μεταβλητες temp\_onoma και temp\_epitheto ως character οπου ειναι arrays των 40 θεσεων και προοριζεται για την προσωρινη αποθηκευση του ονοματος και επιθετου καθε κρατησης που βρισκεται στο txt. Τελος δηλωνουμε αντιστοιχα τις μεταβλητες temp\_tilefwno ως unsigned short και temp\_thesi ως unsigned integer που επισης θα χρησιμοποιουν για την προσωρινη αποθηκευση του τηλεφωνου και του αριθμου θεσης καθε κρατησης που βρισκεται στο txt.

Γραμμη 29: Δηλωση μεταβλητων i και j ως integer που θ χρησιμοποιηθουν ως δεικτες στα for-loop που θα ακολουθησουν.

Γραμμη 33-38: Ανοιγμα αρχειου txt, αποθηκευση του στο leofwreio. Επειτα ελεγχος αν υπαρχει το αρχειο (αν ειναι κενο το lewforeio τοτε δεν υπαρχει το txt και τερματιζει το προγραμμα με exit(1)).

Γραμμη 40: Αποθηκευση πινακιδας και αριθμου θεσεων του λεωφορειου στο pinakida και max

Γραμμη 45-52: Αρχικοποιηση των προσωρινων μεταβλητων των σειρων 26-28

Γραμμη 56-84: Αναγνωση με while-loop του lewforeiou μεχρι το end of file.

Γραμμη 58: προσωρινη αποθηκευση του επιθετου, ονοματος και αριθμου θεσης της κρατησης στα temp\_epitheto, temp\_onoma και temp\_thesi αντιστοιχα

Γραμμη 60-62: προσωρινη αποθηκευση του τηλεφωνου στο ονοματος temp\_tilefwno. Καθε ψηφιο του τελεφωνου αποθηκευεται σε διαφορετικη θεση του temp\_tilefwno (πχ το πρωτο νουμερο στο temp\_tilefwno[0], το επομενο στο temp\_tilefwno[1] κοκ)

Γραμμη 65,66: δινουμε 0 στο i ωστε να ξεκινησει η αριθμηση απο την αρχη και ‘0’ στο found ωστε να ξεκινησει το επομενο while loop.

Γραμμη 67-82: while loop μεχρι να βρεθει σε ποια θεση του πινακα κρατησεων θα αποθηκευτει η κρατηση (οπου οταν βρεθει το found γινεται ισο με 1 και σταματα).

Γραμμη 68-80: Εφοσον το i ειναι ισο temp\_thesi-1 (για αντιστοιχια θεσης πινακα με αριθμο θεσης) αντιγραφουμε τα στοιχεια των μεταβλητων temp\_epitheto και temp\_onoma στο πεδιο onomatepwnupo του thesi[i] βαζοντας ενα κενο αναμεσα. Επειτα στις γραμμες 76-79 με ενα for loop αντιγραφουμεκαθε ψηφιο το temp\_tilefwno στο πεδιο tilefwno του thesi[i]. Τελος αλλαζουμε το found σε ‘1’ ωστε να σταματησει το while loop που ξεκινα απο τη γραμμη 56.

Γραμμη81: αυξανουμε το i κατα ενα ωστε να ελεγξουμε την επομενη θεση του πινακα κρατησεων εαν αντιστοιχει στην επομενη κρατηση του txt.

Γραμμη 85: διαγραφουμε απο την προσωρινη μνημη το lewforeio αφου δεν μας χρειαζεται αλλο.

Γραμμη 90-103: function emfanisi\_kenwn(..) για επιλογη 1 και εμφάνιση συνολικού πλήθους κενών θέσεων και ταξινομημένης λίστας αριθμών τους. Για να γινει αυτο μηδενιζουμε πρωτα το μετρητη kenes.

Γραμμη 93-99: for-loop με το οποιο σκαναρουμε το πινακα κρατησεων απο το μηδεν μεχρι το max. Εαν το πεδιο onomatepwnumo του thesi[i] ειναι ισο με ‘\0’, τοτε επειδη σημαινει οτι ειναι κενη εμφανιζουμε το i αυξημενο κατα 1 ωστε να δωσει τον αριθμο θεσης που αντιστοιχει. Επειτα αυξανουμε το kenes κατα 1

Γραμμη 100-101: Εαν το kenes >0 εμφανιζουμε τη μεταβλητη μιας και αποτελει τον αριθμο των κενων θεσεων. Διαφορετικα εμφανιζουμε μηνυμα οτι δεν υπαρχουν κενες θεσεις.

Γραμμη 106-136: function kratisi\_thesis(..) γα επιλογη 2 για κράτηση θέσης με συγκεκριμένο αριθμό.

Γραμμη 107-110: Δηλωση προσωρινων μεταβλητων temp\_epitheto και temp\_onoma ως character που ειναι arrays των 40 θεσεων. Επειτα ζητα απο το χρηστη να δωσει σε ποια θεση θελει να κανει κρατηση και αποθηκευει την επιλογη του στο i. Επειτα μειωνουμε το i επειδη θα το χρησιμοποιουσουμε ως δεικτη στο πινακα κρατησεων (ετσι θα ειναιι σε αντιστοιχια με τον αριθμο θεσης).

Γραμμη 112-134: **Πρωτα ελεγχουμε** αν η θεση που εδωσε ειναι εντος οριων και κρατημενη (αν ειναι εμφανιζει μηνυμα οτι ειναι κρατημενη και σταματα το function). Επειτα, εφοσον βγει ψευδες το προηγουμενο, **ελεγχουμε** εαν ειναι εντος οριων και ελευθερη (δηλαδη αν onomatepwnumo = ‘\0’). Εαν βγει αληθης, τοτε καταχωρουμε πρωτα τον αριθμο θεσης στο πεδιο arithmos\_thesis του thesi[i] που θα ειναι ισο με i +1 (για να αναιρεσουμε τη προηγουμενη διορθωση). Επειτα καθοδηγει το χρηστη να δωσει επιθετο και ονομα τα οποια τα αποθηκευει στις προσωρινες μεταβλητες temp\_epitheto και temp\_onoma. Μετα στις γραμμες 124-126 μεταφερουμε τα στοιχεια των προσωρινων μεταβλητων στο πεδιο onomatepwnumo του thesi[i]. Επειτα αποθηκευουμε το τηλεφωνο στο πεδιο tilefwno με ενα for-loop. Τελος **εαν εχουν βγει ψευδη και τα 2** προηγουμενα (γραμμη 112 και 115), σημαινει οτι ειναι εκτος οριων του λεωφορειου το νουμερο που εδωσε ο χρηστης και εμφανιζει σχετικο μηνυμα.

Γραμμη 139-190: function anazitisi\_kratisis για επιλογη 3 και αναζητηση κρατησης ειτε με ονοματεπωνυμο ειτε με τηλεφωνο.

Γραμμη 140-145: Δηλωνουμε πρωτα τις προσωρινες μας μεταβλητες για ονοματεπωνυμο οπως και το δεικτη found και match. Το found θα χρησιμοποιηθει ως δεικτης εαν βρεθηκε καποια καταχωρηση με το τηλεφωνο ή ονοματεπωνυμο που εδωσε ο χρηστης, ενω το match θα το χρησιμοποιησουμε στην αναζητηση με τηλεφωνο. Επειτε δηλωνουμε την προσωρινη μεταβλητη για το τηλεφωνο και τελος ζηταμε απο το χρηστη εαν επιθυμει αναζητηση με ονοματεπωνυμο ή με τηλεφωνο και καταχωρουμε την επιλογη του στο i.

Γραμμη 148-166: Εαν δωσει 1 κανουμε αναζητηση με ονοματεπωνυμο. Ζηταμε πρωτα απο το χρηστη να δωσει επιθετο και ονομα και τα αποθηκευουμε στο temp\_epitheto και temp\_onoma αντιστοιχα. Επειτα μεταφερουμε το temp\_onoma στο temp\_epitheto (ωστε να εννιαιο ονοματεπωνυμο) και με ενα for-loop σκαναρουμε το πινακα απο i=0 μεχρι i <max συγκρινοντας το temp\_epitheto με το πεδιο onomatepwnumo. Εαν βγουν «ισα» απο τον ελεγχο με strcmp τοτε ειναι η κρατηση που αναζητουσαμε, εμφανιζουμε τον αριθμο θεσης κανοντας printf το i+1 και αλλαζουμε το found σε ‘0’. Ο ελεγχος γινεται για ολες τις θεσεις καθοτι μπορει να εχει παραπανω απο μια κρατησεις. Αφου τελειωσει το for-loop ελεγχουμε εαν το found = ‘0’. Εαν ναι εμφανιζει μηνυμα οτι δεν υπαρχουν κρατησεις με αυτο το ονοματεπωνυμο.

Γραμμη 169-190: Εαν δωσει 2 κανουμε αναζητηση με τηλεφωνο. Η λογικη που θα ακολουθησουμε ειναι οτι αν εστω ενα στοιχειο του τηλεφωνου που εδωσε ο χρηστης δεν ειναι ισο με το αντιστοιχο του πεδιου tilefwno, τοτε το match αλλαζει σε μηδεν. Αν παλι ολα ειναι ισα τοτε το match παραμεινει ισο με 1 και ειναι αυτο που ψαχναμε.   
Οποτε, ζηταμε πρωτα απο το χρηστη να δωσει το τηλεφωνο και το αποθηκευουμε ψηφιο προς ψηφιο στο temp\_tilefwno. Επειτα στις γραμμες 176-186 με ενα for loop σκαναρουμε το πινακα απο i=0 μεχρι i<max. Κανουμε το match =’1’ και επειτα με ενα «εσωτερικο» for-loop σκαναρουμε τα στοιχειου του τηλεφωνου απο j= 0 μεχρι j<10 και εφοσον το match παραμενει 1 (γραμμες 178-181). Εαν βγει αληθες οτι το temp\_tilefwno[j] και το πεδιο thesi[i].tilefwno[j ] δεν ειναι ισα τοτε αλλαζει το match σε ‘0’ και σταματα αυτο το «εσωτερικο» for-loop. Επειτα ελεγχουμε εαν το match εχει παραμεινει ισο με ‘1’ και αν ειναι αληθες εμφανιζουμε το i+1 που ειναι αριθμος θεσης που ψαχνε ο χρηστης ενω αλλαζουμε το found σε 1. Αφου τελειωσει το for-loop και εμφανισει τυχον κρατησεις, ελεγχουμε εαν το match παρεμεινε ισο με ‘0’ ή αν αλλαξε σε ‘1’. Αν παρεμεινε ‘0’ τοτε δεν βρηκαμε καμια κρατηση και εμφανιζει σχετικο μηνυμα.

Γραμμη 193-217: function akurwsi\_kratisis για επιλογη 4 και ακυρωση της κρατησης της θεσης που επιθυμει ο χρηστης.

Γραμμη 194-199: Δηλωση προσωρινων μεταβλητων για επιθετο και ονομα. Επειτα ζηταμε απο το χρηστη να δωσει τον αριθμο της θεσης που θελει να ακυρωσει την κρατηση, αποθηκευουμε την επιλογη του στο i και επειδη θα το χρησιμοποιησουμε ως δεικτη στο πινακα κρατησεων, το μειωνουμε κατα 1 ωστε να ειναι σε αντιστοιχια με τον αριθμο θεσης.

Γραμμη 201-217: **Ελεγχουμε** εαν ειναι εντος οριων ο αριθμος και αν thesi[i].onomatepwnumo[0] ισουτε με '\0', δηλαδη αν ειναι κενη η θεση. Αν βγει αληθες βγαζει μηνυμα οτι η θεση ειναι ηδη ελευθερη. Αλλιως, στη γραμμη 205 **ελεγχουμε** αν ειναι εντος οριων και οτι η θεση δεν ειναι κενη, δηλαδη οτι thesi[i].onomatepwnumo[0] διαφορο του '\0'. Εαν βγει αληθες κανει το thesi[i].arithmos\_thesis = -1, ολα τα στοιχεια του thesi[i].onomatepwnumo να ειναι ισα με ‘\0’ και «σβηνει» και ολα τα στοιχεια του thesi[i].tilefwno ισο με -1 με τα καταλληλα for-loops. **Εαν παλι δεν βγει αληθες κανενα απο τα προηγουμενα** if των γραμμων 201 και 205, τοτε πιθανοτατα ο χρηστης εδωσε αριθμο εκτος οριων του λεωφορειου και εμφανιζει σχετικο μηνυμα.

Γραμμη 220-233: function emfanisi\_listas(..) για επιλογη 5 και εμφανιση λιστας κρατημενων θεσεων ταξινομημενη κατα αριθμο θεσης.

Γραμμη 223-233: for-loop οπου σκαναρουμε το πινακα κρατησεων απο i=0 μεχρι i<max. Σε καθε θεση του πινακα ελεγχουμε αν ειναι κρατημενη η θεση, δηλαδη αν thesi[i].onomatepwnumo[0] != '\0'. Αν ειναι αληθες τοτε εμφανιζουμε τα στοιχεια της κρατησης με printf τα διαφορα πεδια της.

Γραμμη 236-266: function termatismos\_apothikeusi(..) για αποθηκευση των κρατησεων και τερματισμο του προγραμματος.

Γραμμη 237-247: Δηλωση προσωρινων μεταβλητων για επιθετο, ονομα, τηλεφωνο και αριθμο θεσης. Επειτα περναω το txt στο lewforeio οπου αν ειναι κενο τοτε δεν υπαρχει το αρχειο txt που καλεσαμε και τερματιζει. Στη γραμμη 250 περναω την πινακιδα και το max αριθμο θεσεων του λεωφορειου. Επειτα στις γραμμες 253-262 αποθηκευω τις κρατησεις με ενα for-loop. Συγκεκριμενα ελεγχω πρωτα αν το thesi[i].onomatepwnumo[0] ειναι διαφορο του ‘\0’, δηλαδη αν η θεση ειναι κρατημενη. Αν ειναι αληθες αλλαζω σειρα στο lewforeio και περναω τα στοιχεια ολων των πεδιων του thesi[i]. Αυτο επαναλαμβανεται μεχρι να ελεγξει και τη τελευταια θεση του thesi[i]. Οταν τελειωσει το for-loop σβηνουμε απο τη μνημη το lewforeio και εμφανιζει μηνυμα στο χρηστη οτι ζητησε εξοδο και αποθηκευτηκαν τα στοιχεια στο αρχειο bus.txt.

Γραμμη 268-302: το main του προγραμματος. (απο εδω ξεκιναει).

Γραμμη 269-282: Δηλωνουμε το πινακα thesi[53] βασει της δομης EPIBATIS που δηλωσαμε στις γραμμες 6-10 ενω δηλωνουμε το pinakida τυπου character ως pointer οπως επισης και το max τυπου integer παλι ως pointer. Αυτες οι μεταβλητες θα χρησιμοποιηθουν στα function με call-by-reference. Ακομη δηλωνουμε το kenes, I, j, και epilogi ως integers οπου το πρωτο θα χρησιμοποιηθει για την μετρηση των κενων θεσεων, τα i και j ως δεικτες σε διαφορα loops και πινακες και το epilogi ως δεικτης για την επιλογη του χρηστη. Να σημειωθει οτι αρχικοποιουμε το epilogi = -1 για να ξεκινησει το while loop στη γραμμη 288. Τελος δεσμευουμε μνημη για το max και pinakida ενω αρχικοποιουμε τα πεδια του thesi ωστε και οι 53 θεσεις να ειναι κενες.

Γραμμη 284-286: καλουμε το function read\_file για αναγνωση του txt. Προκειμενου να τα τραβηξουμε στο main περναμε τις μεταβλητες με cal by reference. Επειτα απλα εμφανιζουμε την πινακιδα και το μεγιστο αριθμο θεσεων του λεωφορειου.

Γραμμη 288-300: while-loop που επαναλαμβανεται μεχρι ο χρηστης να ζητησει εξοδο δινοντας 0.

Γραμμη 289-291: Εμφανιση των διαθεσιμων επιλογων και αποθηκευση της επιλογης του χρηστη στο epilogi.

Γραμμη 293-298: Αναλογως αν ο χρηστης δωσει 0 ως 5 τρεχει το termatismos\_apothikeusi, function emfanisi\_kenwn, kratisi\_thesis, anazitisi\_kratisis, akurwsi\_kratisis και emfanisi\_listas αντιστοιχα. Στο function termatismos\_apothikeusi περναμε το thesi, \*pinakida, \*max, i, και j. Σε ολα τα υπολοιπα περναμε τα thesi, \*max, i, και j εκτος απο το function emfanisi\_kenwn οπου αντι για το j βαζουμε το kenes. Το \*max το βαζουμε με αστερακι γιατι σε αντιθεση με το read\_file στη γραμμη 284 εδω μας αρκει το call by value, οποτε με το αστερακι τραβαμε την τιμη που δειχνει. Τελος, εαν δεν επιλεχθει κανενα απο το 0 εως 5 εμφανιζει μηνυμα οτι δεν υπαρχει η δραστηριοτητα που ζητησε.

# Θεμα 2

Γραμμη 6-10: Το structure που θα χρησιμοποιηθει για την αναπαρασταση θεσεων με πεδια onomatepwnumo ως character που ειναι array 40 θεσεων για την αποθηκευση του ονοματεπωνυμου, arithmos\_thesis ως unsigned integer για τον αριθμο θεσης και tilefwno ως insigned short που ειναι array 10 θεσεων.

Γραμμη 13-18: Το αντιστοιχο structure για linked list, με ιδια πεδια με το προηγουμενο με επιπλεον το πεδιο next που δειχνει τη θεση μνημης του επομενου node.

Γραμμη 21-28: function menu() που εμφανιζει με αριθμημενη λιστας τις δυνατες δραστηριοτητες που μπορει να επιλεξει ο χρηστης.

Γραμμη 31-48: function initialize(..) για αναγνωση και αποθηκευση του μεγιστου αριθμου θεσεων του λεωφορειου.

Γραμμη 32: δηλωση μεταβλητης temp\_pinakida ως character που ειναι array των 8 θεσεων.

Γραμμη 35-41: ανοιγμα αρχειου txt, αποθηκευση του στο leofwreio. Τελος κανουμε ελεγχο αν υπαρχει το αρχειο (αν ειναι κενο το lewforeio τοτε δεν υπαρχει το txt και τερματιζει το προγραμμα με exit(1) αφου εμφανισει σχετικο μηνυμα).

Γραμμη 44: Αποθηκευση πινακιδας και αριθμου θεσεων του λεωφορειου στο temp\_pinakida και στο max

Γραμμη 46: διαγραφουμε απο την προσωρινη μνημη το lewforeio αφου δεν μας χρειαζεται αλλο.

Γραμμη 50-102: function read\_file(..) για αναγνωση και αποθηκευση των στοιχειων του λεοφωρειου απο txt.

Γραμμη 51-53: δηλωση μεταβλητης found ως character που προοριζεται για ελεγχο στο for-loop γραμμη 81-95 οπου σκαναρουμε τον πινακα των θεσεων για το που θα αποθηκευτει η καθε κρατηση καθως και το temp\_max παλι ως character. Ακομη δηλωνουμε και τις μεταβλητες temp\_onoma και temp\_epitheto ως character οπου ειναι arrays των 40 θεσεων και προοριζεται για την προσωρινη αποθηκευση του ονοματος και επιθετου καθε κρατησης που βρισκεται στο txt. Τελος δηλωνουμε αντιστοιχα τις μεταβλητες temp\_tilefwno ως unsigned short και temp\_thesi ως unsigned integer που επισης θα χρησιμοποιουν για την προσωρινη αποθηκευση του τηλεφωνου και του αριθμου θεσης καθε κρατησης που βρισκεται στο txt.

Γραμμη 54: Δηλωση μεταβλητων i και j ως integer που θ χρησιμοποιηθουν ως δεικτες στα for-loop που θα ακολουθησουν.

Γραμμη 57-63: Ανοιγμα αρχειου txt, αποθηκευση του στο leofwreio. Επειτα ελεγχος αν υπαρχει το αρχειο (αν δεν υπαρχει τερματιζει το προγραμμα με exit(1).

Γραμμη 67: Αποθηκευση πινακιδας και αριθμου θεσεων του λεωφορειου στο pinakida και temp\_max

Γραμμη 71-97: Αναγνωση του lewforeiou με while-loop μεχρι το end of file.

Γραμμη 72: προσωρινη αποθηκευση του επιθετου, ονοματος και αριθμου θεσης της κρατησης στα temp\_epitheto, temp\_onoma και temp\_thesi αντιστοιχα

Γραμμη 74-77: προσωρινη αποθηκευση του τηλεφωνου στο ονοματος temp\_tilefwno. Καθε ψηφιο του τελεφωνου αποθηκευεται σε διαφορετικη θεση του temp\_tilefwno (πχ το πρωτο νουμερο στο temp\_tilefwno[0], το επομενο στο temp\_tilefwno[1] κοκ). Το τελευταιο ψηφιο το διαβαζουμε εκτος του loop ωστε να διαβασει και το χαρακτηρα της αλλαγης σειρας (δηλαδη το ‘\n’)

Γραμμη 80: δινουμε ‘0’ στο found ωστε να ξεκινησει το επομενο for- loop.

Γραμμη 81-95: for loop οπου σκαναρει το πινακα απο i=0 εως i<temp\_max μεχρι να βρεθει σε ποια θεση του πινακα κρατησεων θα αποθηκευτει η κρατηση (οπου οταν βρεθει το found γινεται ισο με 1 και σταματα).

Γραμμη 83-93: Εφοσον το thesi[i].arithmos\_thesis ειναι ισο temp\_thesi αντιγραφουμε τα στοιχεια των μεταβλητων temp\_epitheto και temp\_onoma στο πεδιο onomatepwnupo του thesi[i] βαζοντας ενα κενο αναμεσα. Επειτα στις γραμμες 89-91 με ενα for loop αντιγραφουμεκαθε ψηφιο το temp\_tilefwno στο πεδιο tilefwno του thesi[i]. Τελος αλλαζουμε το found σε ‘1’ ωστε να σταματησει το while loop που ξεκινα απο τη γραμμη 81.

Γραμμη 100: διαγραφουμε απο την προσωρινη μνημη το lewforeio αφου δεν μας χρειαζεται αλλο.

Γραμμη 105-118: function emfanisi\_kenwn(...) για επιλογη 1 και εμφάνιση συνολικού πλήθους κενών θέσεων και ταξινομημένης λίστας αριθμών τους. Για να γινει αυτο μηδενιζουμε πρωτα το μετρητη kenes.

Γραμμη 108-114: for-loop με το οποιο σκαναρουμε το πινακα κρατησεων απο το μηδεν μεχρι το max. Εαν το πεδιο onomatepwnumo του thesi[i] ειναι ισο με ‘\0’, τοτε επειδη σημαινει οτι ειναι κενη εμφανιζουμε τη τιμη του πεδιου arithmos\_thesis. Επειτα αυξανουμε το kenes κατα 1

Γραμμη 115,116: Εαν το kenes >0 εμφανιζουμε τη μεταβλητη μιας και αποτελει τον αριθμο των κενων θεσεων. Διαφορετικα εμφανιζουμε μηνυμα οτι δεν υπαρχουν κενες θεσεις.

Γραμμη 121-151: function kratisi\_thesis(..) γα επιλογη 2 για κράτηση θέσης με συγκεκριμένο αριθμό.

Γραμμη 122-125: Δηλωση προσωρινων μεταβλητων temp\_epitheto και temp\_onoma ως character που ειναι arrays των 40 θεσεων. Επειτα ζητα απο το χρηστη να δωσει σε ποια θεση θελει να κανει κρατηση και αποθηκευει την επιλογη του στο i. Επειτα μειωνουμε το i επειδη θα το χρησιμοποιουσουμε ως δεικτη στο πινακα κρατησεων (ετσι θα ειναιι σε αντιστοιχια με τον αριθμο θεσης).

Γραμμη 127-149: **Πρωτα ελεγχουμε** αν η θεση που εδωσε ειναι εντος οριων και κρατημενη (αν ειναι εμφανιζει μηνυμα οτι ειναι κρατημενη και σταματα το function). Επειτα, εφοσον βγει ψευδες το προηγουμενο, **ελεγχουμε** εαν ειναι εντος οριων και ελευθερη (δηλαδη αν onomatepwnumo = ‘\0’). Εαν βγει αληθης, τοτε καταχωρουμε πρωτα τον αριθμο θεσης στο πεδιο arithmos\_thesis του thesi[i] που θα ειναι ισο με i +1 (για να αναιρεσουμε τη προηγουμενη διορθωση). Επειτα καθοδηγει το χρηστη να δωσει επιθετο και ονομα τα οποια τα αποθηκευει στις προσωρινες μεταβλητες temp\_epitheto και temp\_onoma. Μετα στις γραμμες 139-141 μεταφερουμε τα στοιχεια των προσωρινων μεταβλητων στο πεδιο onomatepwnumo του thesi[i]. Επειτα αποθηκευουμε το τηλεφωνο στο πεδιο tilefwno με ενα for-loop. Τελος στη γραμμη 149, **εαν εχουν βγει ψευδη και τα 2** προηγουμενα (γραμμη 127 και 130), σημαινει οτι ειναι εκτος οριων του λεωφορειου το νουμερο που εδωσε ο χρηστης και εμφανιζει σχετικο μηνυμα.

Γραμμη 154-205: function anazitisi\_kratisis για επιλογη 3 και αναζητηση κρατησης ειτε με ονοματεπωνυμο ειτε με τηλεφωνο.

Γραμμη 155-160: Δηλωνουμε πρωτα τις προσωρινες μας μεταβλητες για ονοματεπωνυμο οπως και το δεικτη found και match. Το found θα χρησιμοποιηθει ως δεικτης εαν βρεθηκε καποια καταχωρηση με το τηλεφωνο ή ονοματεπωνυμο που εδωσε ο χρηστης, ενω το match θα το χρησιμοποιησουμε στην αναζητηση με τηλεφωνο. Επειτε δηλωνουμε την προσωρινη μεταβλητη για το τηλεφωνο και τελος ζηταμε απο το χρηστη εαν επιθυμει αναζητηση με ονοματεπωνυμο ή με τηλεφωνο και καταχωρουμε την επιλογη του στο i.

Γραμμη 163-181: Εαν δωσει 1 κανουμε αναζητηση με ονοματεπωνυμο. Ζηταμε πρωτα απο το χρηστη να δωσει επιθετο και ονομα και τα αποθηκευουμε στο temp\_epitheto και temp\_onoma αντιστοιχα. Επειτα μεταφερουμε το temp\_onoma στο temp\_epitheto (ωστε να εννιαιο ονοματεπωνυμο) και με ενα for-loop σκαναρουμε το πινακα απο i=0 μεχρι i <max συγκρινοντας το temp\_epitheto με το πεδιο onomatepwnumo. Εαν βγουν «ισα» απο τον ελεγχο με strcmp τοτε ειναι η κρατηση που αναζητουσαμε, εμφανιζουμε τον αριθμο θεσης κανοντας printf το thesi[i].arithmos\_thesis και αλλαζουμε το found σε ‘0’. Ο ελεγχος γινεται για ολες τις θεσεις καθοτι μπορει να εχει παραπανω απο μια κρατησεις. Αφου τελειωσει το for-loop ελεγχουμε εαν το found = ‘0’. Εαν ναι εμφανιζει μηνυμα οτι δεν υπαρχουν κρατησεις με αυτο το ονοματεπωνυμο.

Γραμμη 184-203: Εαν δωσει 2 κανουμε αναζητηση με τηλεφωνο. Η λογικη που θα ακολουθησουμε ειναι οτι αν εστω ενα στοιχειο του τηλεφωνου που εδωσε ο χρηστης δεν ειναι ισο με το αντιστοιχο του πεδιου tilefwno, τοτε το match αλλαζει σε μηδεν. Αν παλι ολα ειναι ισα τοτε το match παραμεινει ισο με 1 και ειναι αυτο που ψαχναμε.   
Οποτε, ζηταμε πρωτα απο το χρηστη να δωσει το τηλεφωνο και το αποθηκευουμε ψηφιο προς ψηφιο στο temp\_tilefwno. Επειτα στις γραμμες 191-201 με ενα for loop σκαναρουμε το πινακα απο i=0 μεχρι i<max. Κανουμε το match =’1’ και επειτα με ενα «εσωτερικο» for-loop σκαναρουμε τα στοιχειου του τηλεφωνου απο j= 0 μεχρι j<10 και εφοσον το match παραμενει 1 (γραμμες 193-196). Εαν βγει αληθες οτι το temp\_tilefwno[j] και το πεδιο thesi[i].tilefwno[j ] δεν ειναι ισα τοτε αλλαζει το match σε ‘0’ και σταματα αυτο το «εσωτερικο» for-loop. Επειτα ελεγχουμε εαν το match εχει παραμεινει ισο με ‘1’ και αν ειναι αληθες εμφανιζουμε το i+1 που ειναι αριθμος θεσης που ψαχνε ο χρηστης ενω αλλαζουμε το found σε 1. Αφου τελειωσει το for-loop και εμφανισει τυχον κρατησεις, ελεγχουμε εαν το match παρεμεινε ισο με ‘0’ ή αν αλλαξε σε ‘1’. Αν παρεμεινε ‘0’ τοτε δεν βρηκαμε καμια κρατηση και εμφανιζει σχετικο μηνυμα.

Γραμμη 208-231: function akurwsi\_kratisis για επιλογη 4 και ακυρωση της κρατησης της θεσης που επιθυμει ο χρηστης.

Γραμμη 209-214: Δηλωση προσωρινων μεταβλητων για επιθετο και ονομα. Επειτα ζηταμε απο το χρηστη να δωσει τον αριθμο της θεσης που θελει να ακυρωσει την κρατηση, αποθηκευουμε την επιλογη του στο i και επειδη θα το χρησιμοποιησουμε ως δεικτη στο πινακα κρατησεων, το μειωνουμε κατα 1 ωστε να ειναι σε αντιστοιχια με τον αριθμο θεσης.

Γραμμη 216-229: **Ελεγχουμε** εαν ειναι εντος οριων ο αριθμος και αν thesi[i].onomatepwnumo[0] ισουτε με '\0', δηλαδη αν ειναι κενη η θεση. Αν βγει αληθες βγαζει μηνυμα οτι η θεση ειναι ηδη ελευθερη. Αλλιως, στη γραμμη 220 **ελεγχουμε** αν ειναι εντος οριων και οτι η θεση δεν ειναι κενη, δηλαδη οτι thesi[i].onomatepwnumo[0] διαφορο του '\0'. Εαν βγει αληθες κανει τα στοιχεια του thesi[i].onomatepwnumo να ειναι ισα με ‘\0’ και «σβηνει» και ολα τα στοιχεια του thesi[i].tilefwno ισο με -1 με τα καταλληλα for-loops. **Εαν παλι δεν βγει αληθες κανενα απο τα προηγουμενα** if των γραμμων 216 και 220, τοτε πιθανοτατα ο χρηστης εδωσε αριθμο εκτος οριων του λεωφορειου και εμφανιζει σχετικο μηνυμα.

Γραμμη 234-330: function emfanisi\_listas(..) για επιλογη 5 και εμφανιση λιστας κρατημενων θεσεων ταξινομημενη ειτε κατα αριθμο θεσης ειτε κατα αλφαβητικη σειρα.

Γραμμη 235-242: Δηλωση μεταβλητης swap ως integer για ελεγχο εαν εγινε αλλαγη στο σορταρισμα και την αρχικοποιουμε ισο με 1 ωστε να ξεκινησει το while-loop στη γραμμη 262. Ακομη δηλωνουμε τις προσωρινες μεταβλητες για τα στοιχεια των κρατησεων. Ρωταμε το χρηστη πως τις θελει ταξινομημενες και καταχωρουμε την επιλογη του στο i.

Γραμμη 246-257: Εαν εδωσε 1 τοτε με ενα for-loop σκαναρουμε το πινακα κρατησεων απο i=0 μεχρι i<max και ελεγχουμε καθε θεση του πινακα αν ειναι κρατημενη η θεση, δηλαδη αν thesi[i].onomatepwnumo[0] != '\0'. Αν ειναι αληθες τοτε εμφανιζουμε τα στοιχεια της κρατησης με printf τα διαφορα πεδια της. (δεν χρειαζεται σορταρισμα μιας και ειναι ηδη ταξινομημενος κατα αριθμο θεσης)

Γραμμη 260-329: Εαν εδωσε 2 τοτε ο χρηστης θελει ταξινομηση κατα ονοματεπωνυμο και κανουμε τα ακολουθα:

Γραμμη 262-287: ξεκιναμε while-loop για ταξινομηση μεχρι το swap να ειναι ισο με το μηδεν (δηλαδη σταματαει οταν δεν ανταλλαξε καμια θεση). Κανουμε το swap ισο με το μηδεν και με ενα for-loop σκαναρουμε τον πινακα κρατησεων απο το i=0 μεχρι i<max ελεγχοντας αν η κρατηση i+1 επεται αλφαβητικα ωστε να ανταλλαξουμε τα στοιχεια τους. Πιο συγκεκριμενα στη γραμμη 265 με ενα strcmp ελεγχουμε αν το περιεχομενο του πεδιου onomatepwnumo της θεσης i επεται του αντιστοιχου της θεσης i+1. Εαν ναι προχωραμε σε ανταλλαγη των στοιχειων τους αποθηκευοντας πρωτα το onomatepwnumo του i+1 στο προσωρινο temp\_onomatepwnumo. Επειτα το onomatepwnumo του i+1 παιρνει το περιεχομενο του onomatepwnumo απο το i. Εν συνεχεια το onomatepwnumo του i παιρνει το περιεχομενο απο το προσωρινο temp\_onomatepwnumo. Παρομοια διαδικασια επαναλαμβανεται για το τηλεφωνο και τον αριθμο θεσης. Τελος αλλαζουμε το swap σε 1 ωστε οταν τελειωσει το for-loop να επαναλυφθει το while-loop για τυχον αναγκαιες νεες αλλαγες.

Γραμμη 292-301: με ενα for-loop σκαναρουμε τομ ταξινομημενο μας πλεον πινακα κρατησεων απο i=0 μεχρι i<max και ελεγχουμε καθε θεση του πινακα αν ειναι κρατημενη η θεση, δηλαδη αν thesi[i].onomatepwnumo[0] != '\0'. Αν ειναι αληθες τοτε εμφανιζουμε τα στοιχεια της κρατησης με printf τα διαφορα πεδια της.

Γραμμη 304-329: θα επαναφερουμε τον πινακα στην αρχικη κατασταση, δηλαδη ταξινομημενο κατα αριθμο θεσης. Ετσι κανουμε το swap ισο με 1 για να ξεκινησει το επομενο while-loop. Ειναι ιδια διαδικασια με τη ταξινομηση κατα αλφαβητικη μονο που στο for-loop αντι να ελεγχουμε αν επεται αλφαβητικα μια καταχωρηση, ελεγχουμε αν επεται κατα αριθμο θεσης. Δηλαδη κανουμε ελεγχο thesi[i].arithmos\_thesis > thesi[i+1].arithmos\_thesis και μετα ακολουθουμε τα ιδια βηματα.

Γραμμη 333-421: function termatismos\_apothikeusi(..) για αποθηκευση των κρατησεων και τερματισμο του προγραμματος.

Γραμμη 334-338: Δηλωση των epibatis, tmp και start ως struct της μορφης που δηλωθηκε στις γραμμες 13-18. Στο epibatis θα αποθηκευονται τα στοιχεια της τρεχουσας κρατησης που θελουμε να περασουμε και μετα «κουμπωνουμε» τα στοιχεια του στο τελος του tmp που θα ειναι η linked list μας. Το start θα χρησιμοποιηθει ως δεικτης του που ξεκινα η linked list. Επισης δηλωνουμε τις προσωρινες μεταβλητες για επιθετο, ονομα, τηλεφωνο και αριθμο θεσης. Τελος δηλωνουμε το swap ως integer και ισο με 1 που θα χρησιμοποιηθει στη ταξινομηση κατα ονοματεπωνυμο στο while-loop που ακολουθει (γραμμες 341 – 362).

Γραμμη 341-362: while-loop για ταξινομηση αλφαβητικα. Ειναι ιδια ακριβως διαδικασια που ακολουθησαμε παραπανω στις γραμμες 262-287 (ενδεχομενως θα μπορουσαμε να χαμε κανει ενα ξεχωριστο function και να το καλουμε).

Γραμμη 366-391: for-loop για σκαναρισμα του πινακα κρατησεων. Εαν υπαρχει κρατηση στο i, δηλαδη αν thesi[i].onomatepwnumo[0] != '\0', τοτε προχωραμε ως εξης. Αποθηκευουμε τα στοιχεια της στο epibatis. Για να γινει ομως αυτο πρεπει πρωτα με malloc να δεσμευσουμε μνημη ιση με το μεγεθος του struct μας, δηλαδη του Node .Επειτα κανουμε το πεδιο next να ειναι ισο με NULL και αντιγραφουμε τα στοιχεια της κρατησης στα αντιστοιχα πεδια του epibatis. Αφου ολοκληρωθει περναμε τα στοιχεια του epibatis στη linked list μας μεσω των γραμμων 382-387 οπου ελεγχουμε πρωτα αν το start ειναι ισο με NULL. Εαν ναι το κανουμε ισο με το epibatis προκειμενου να ξερουμε απο που ξεκινα η linked list μας. Εαν οχι τοτε το προσθετουμε στο τελος της Linked list μας (προχωραμε το tmp κατα μια θεση και μετα βαζουμε το next να δειχνει το epibatis που μολις καταχωρησαμε).

Γραμμη 395-419: συνδεω το txt με το lewforeio οπου αν ειναι κενο τοτε δεν υπαρχει το αρχειο txt που καλεσαμε και τερματιζει. Στη γραμμη 404 περναω την πινακιδα και το max αριθμο θεσεων του λεωφορειου. Επειτα στη γραμμη 408 ξεκιναω την linked list απο το αρχικο node (θετοντας το ισο με start) και επειτα με ενα while-loop αποθηκευω τις κρατησεις που σταματα οταν το next = NULL οπου θα ειναι και η τελευταια κρατηση. Συγκεκριμενα περναω με ενα fprintf τα διαφορα πεδια του tmp στο lewforeio και προχωραω στη λιστα μου κατα ενα node βαζοντας το tmp=tmp->next. Οταν τελειωσει το while-loop σβηνουμε απο τη μνημη το lewforeio και εμφανιζει μηνυμα στο χρηστη οτι ζητησε εξοδο και αποθηκευτηκαν τα στοιχεια στο αρχειο bus\_thema2.txt.

Γραμμη 423-466: το main του προγραμματος. (απο εδω ξεκιναει).

Γραμμη 424-431: Δηλωνουμε το πινακα thesi[53] ως structure EPIBATIS που δηλωσαμε στις γραμμες 6-10 ενω δηλωνουμε το pinakida τυπου character ως pointer οπως επισης και το max τυπου integer παλι ως pointer. Αυτες οι μεταβλητες θα χρησιμοποιηθουν στα functions με call-by-reference. Ακομη δηλωνουμε το kenes, I, j, και epilogi ως integers οπου το πρωτο θα χρησιμοποιηθει για την μετρηση των κενων θεσεων, τα i και j ως δεικτες σε διαφορα loops και πινακες και το epilogi ως δεικτης για την επιλογη του χρηστη. Να σημειωθει οτι αρχικοποιουμε το epilogi = -1 για να ξεκινησει το while loop στη γραμμη 449. Τελος δεσμευουμε μνημη για το max και pinakida.

Γραμμη 434-446: με το initialize\_arithmos\_thesewn τραβαμε το μεγιστο αριθμο θεσεων που χωρα το λεωφορειο με call-by-reference το max. Μετα δεσμευουμε μνημη για το thesi ωστε να χωρα max φορες το μεγεθος του structure ενω με ενα for-loop απο i=0 εως i<max αρχικοποιουμε τα πεδια του thesi ωστε να ειναι ελευθερες (onomatepwnumo[0] =’\0’) και αριθμημενες (δηλαδη arithmos\_thesis = i+1). Επειτα καλουμε το function read\_file για να τραβηξουμε στο main και τα υπολοιπα στοιχεια παλι με call by reference. Τελος εμφανιζουμε την πινακιδα και το μεγιστο αριθμο θεσεων του λεωφορειου.

Γραμμη 449-464: while-loop που επαναλαμβανεται μεχρι ο χρηστης να ζητησει εξοδο δινοντας 0.

Γραμμη 451-453: Εμφανιση των διαθεσιμων επιλογων και αποθηκευση της επιλογης του χρηστη στο epilogi.

Γραμμη 456-462: Αναλογως αν ο χρηστης δωσει 0 ως 5 τρεχει το termatismos\_apothikeusi, function emfanisi\_kenwn, kratisi\_thesis, anazitisi\_kratisis, akurwsi\_kratisis και emfanisi\_listas αντιστοιχα. Στο function termatismos\_apothikeusi περναμε το thesi, \*pinakida, \*max, i, και j. Σε ολα τα υπολοιπα 1-5 περναμε τα thesi, \*max, i, και j εκτος απο το function emfanisi\_kenwn οπου αντι για το j βαζουμε το kenes. Το \*max και \*pinakida τα βαζουμε με αστερακι γιατι σε αντιθεση με το initialize\_arithmos\_thesewn και read\_file στις γραμμες 434 και 444 αντιστοιχα, μας αρκει call by value, οποτε με το αστερακι τραβαμε το περιεχομενο που δειχνουν. Τελος, εαν δεν επιλεχθει κανενα απο το 0 εως 5 εμφανιζει μηνυμα οτι δεν υπαρχει η δραστηριοτητα που ζητησε.

Παρατηρησεις για θεμα 3

1.Στο θεμα 3 χρησιμοποιησα ενα επιπλεον πεδιο στη δομη του KARTES που περιεχει ως μια λεξη ολα τα χαρακτηριστικη της καρτας προκειμενου να ειναι ευκολος ο ελεγχος εαν υπαρχει διπλη καρτα μεσω ενος απλου strcmp αυτου του πεδιου. Ακομη ενω θα μπορουσα (και οντως ετσι ειχα κανει στην αρχη) να εμφανιζω απλα αυτο το πεδιο οταν εμφανιζα τις καρτες τελικα επελεξα να εμφανισω τα 4 πεδια με ενα κομμα αναμεσα ωστε να εχει ιδια μορφη με αυτη που προτεινετε στην ασκηση.

2. Εχω βαλει μεσα και ενα επιπλεον function το cheat για ελεγχο αν υπαρχει τριαδα set και επιστροφη ή οχι των συντεταγμενων τους. Μπορει να μην το ζηταγε η ασκηση αλλα ειχα 2 πλεονεκτηματα με αυτο.

Α. Εκανα εναν ελεγχο αν υπαρχει εστω και ενα δυνατο σετ (δεν ξερω αν οντως χρειαζοταν μιας και δεν προχωρησα σε υπολογισμο των πιθανοτητων αν ειναι εφικτο να μην υπαρχει σετ). Οποτε εαν δεν υπαρχει κανενα δυνατο σετ τοτε το προγραμμα τραβαει νεες καρτες.

Β. Εχω αφησει ενα κρυφο cheat mode οπου αν ο καποιος δωσει C εμφανιζει μεσω του function μια τριαδα καρτων που κανουν σετ. Αυτο αφενως ειναι μια επιλογη που θα μπορουσε να ειχε το παιχνιδι και αφετερου βοηθα και εμας να ελεγξουμε ευκολα αν δουλευει σωστα το προγραμμα (βλεπουμε ποια τριαδα κανει σετ και την βαζουμε για να προχωρησει)

3. Οποτε καλουμαστε να αυξησουμε το σκορ κατα 2 μοναδες ελεγχουμε εαν δεν εχει ξεπερασει το 255 που ζητα η ασκηση στην αρχη. Ισως ειναι περιττο γιατι εαν συγκεντρωσει 10 βαθμους το προγραμμα τερματιζει

4. Εχω ξεκινησει το πρωτο while-loop να επαναλαβανεται οσο το epilogi != 0. Ισως βεβαια ειναι περιττο να βαλουμε το οποιδηποτε variable και απλα βαζαμε τον αριθμο 1 ωστε να επαναλαμβανεται ες αει μιας και οταν δωσει ο χρηστης 0 τοτε καλουμε το exit(1) και οποτε τερματιζει το προγραμμα.

# Θεμα 3

Γραμμη 6-9: δηλωση δομης με τα χαρακτηριστικα των καρτων. Συγκεκριμενα δηλωνουμε τα πεδια xrwma, sxima, arithmos, ufh ως character που ειναι arrays 2 στοιχειων. Ακομη δηλωνουμε το enniaio ως character που ειναι array 5 στοιχειων και θα περιλαμβανει εννιαια ολα τα στοιχεια της καρτας (πχ για κοκκινο, κυκλο, 3 και πληρες θα εχει περιεχομενο rc3b)

Γραμμη 12-16: δηλωση δομης για τα στοιχεια των παικτων. Συγκεκριμενα δηλωνουμε τα πεδια id ως character, onoma ως character που ειναι array 17 στοιχειων και score ως unsigned integer.

Γραμμη 18-85: function test\_if\_set(..) για ελεγχο εαν μια τριαδα καρτων ειναι σετ. Αρχικα θετουμε το set ισο με ‘0’ και το point = 0. Για καθε χαρακτηριστικο (πχ χρωμα) ελεγχουμε εαν ειναι σε ολες τις καρτες ιδιο ή σε ολες τις καρτες διαφορετικο. Εαν ισχυει ενα απο τα δυο τοτε αυξανουμε το point κατα μια μοναδα και προχωραμε στο επομενο χαρακτηριστικο (πχ σχημα) και κανουμε την ιδια διαδικασια. Οταν τελειωσουμε με ολα τα χαρακτηριστικα εαν συγκεντρωσαμε point=4 σημαινει οτι καναμε σετ. Με αυτη τη λογικη λοιπον ξεκιναμε αναλυτικα:

Γραμμη 25-29: ελεγχουμε με strcmp αν το xrwma της πρωτης καρτας ειναι ιδιο με το xrwma της δευτερης καρτας (τις καρτες τις βρισκουμε μεσω των συντεταγμενων που εδωσε ο χρηστης στο πινακα karta[][]). Εαν ειναι ισο προχωραμε και ελεγχουμε εαν το χρωμα της τριτης καρτας ειναι ιδιο με της δευτερης. Εαν βγει και αυτο αληθες (οποτε ειναι ολα ιδια) τοτε αυξανουμε το point κατα 1.

Γραμμη 31-37: εαν βγει ψευδες το προηγουμενο τοτε ελεγχουμε με strcmp αν το xrwma της πρωτης καρτας ειναι διαφορετικο απο το xrwma της δευτερης καρτας. Εαν ειναι διαφορετικο προχωραμε και ελεγχουμε εαν το χρωμα της τριτης καρτας ειναι διαφορετικο με της δευτερης. Εαν ειναι διαφορετικο ελεγχουμε εαν το χρωμα της τριτης καρτας ειναι διαφορετικο απο της πρωτης. Εαν βγει και αυτο αληθες (οποτε ειναι ολα διαφορετικα) τοτε αυξανουμε το point κατα 1.

Γραμμη 39-79: κανουμε ιδια διαδικασια με γραμμες 25-37 για τα χαρακτηριστικα sxima, arithmos και στηλη

Γραμμη 82-84: ελεγχουμε εαν συγκεντρωσαμε 4 στο point. Αν βγει αληθες τοτε αλλαζουμε το set σε ‘1’. Τελος κανουμε return το set

Γραμμη 90-93: function getRandomNumber(..) για να τραβηξουμε τυχαιους αριθμους. Τραβαει αριθμους απο min μεχρι max που δινει ο χρηστης.

Γραμμη 97-121: function getNextCard(..) για να τραβηξουμε μια καρτα. Πρωτα δηλωνουμε το card τυπου structure της μορφης KARTES που δηλωσαμε στις γραμμες 6-9. Επειτα θετουμε το στοιχειο 0 του πεδιου arithmos να ειναι ισο με ενα στοιχειο απο το πινακα set\_arithmoi. Ο πινακας set\_arithmoi εχει αρχικοποιηθει στο main και τα στοιχεια του απο 0 ως 2 περιεχουν τιμες 1 ως 3 αντιστοιχα. Για να τραβηξουμε μια τυχαια τιμη απο αυτο το πινακα βαζουμε ως δεικτη μεσα στο πινακα το getRandomNumber(0,2). Ετσι ο τυχαιος δεικτης που θα τραβηξουμε θα καταληξει σε τυχαια τιμη απο το set\_arithmoi. Ισως για το χαρακτηριστικο arithmoi να μπορουσαμε να τραβηξουμε απευθειας τιμες χωρις τη βοηθεια πινακα. Ομως αυτη η μεθοδολογια εχει σημασια για τα υπολοιπα χαρακτηριστικα οπου τραβαμε γραμματα και οχι αριθμους. Ετσι τα χαρακτηριστικα xrwma, sxima και ufh τραβανε τυχαια στοιχεια απο τους πινακες set\_xrwmata, set\_sximata και set\_ufes αντιστοιχα μεσω του τυχαιου δεικτη getRandomNumber(0,2) σε καθε πινακα. Αφου ολοκληρωθουν τα παραπανω, βαζουμε στη θεση 1 του καθε πεδιου να ειναι ισο με ‘\0’. Ο λογος που καναμε δυσδιαστατους πινακες ειναι προκειμενου να μπορουν να χρησιμοποιηθουν οι εντελος strcmp και strcpy οπυο ειναι αναγκαιο να γνωριζει το προγραμμα που τελειωνει το string (αρα ειναι αναγκαιο το ‘\0’). Τελος γεμιζουμε τα στοιχεια απο 0 εως 3 του πεδιου enniaio με το περιεχομενο των πεδιων xrwma[0], sxima[0], arithmos[0] και ufh[0] αντιστοιχα. Τελος οριζουμε το enniaio[4]='\0' για το τελος του string και κανουμε return τo card.

Γραμμη 123-139: function cheat(..) για να βρουμε μια τριαδα καρτων που κανει σετ. Θα χρειαστει. Η λογικη ειναι να βρισκουμε πρωτα μια καρτα (μεσω διπλου for-loop για τις συντεταγμενες). Επειτα σκαναρουμε αντιστοιχα για δευτερη καρτα ολες τις υπολοιπες που ειναι μετα απο αυτην. Τελος για καθε δευτερη που ελεχουμε σκαναρουμε ολες τις υπολοιπες καρτες μετα την δευτερη. Η διαδικασια με αυτο το 6πλο for-loop προχωραει μεχρι με το test\_if\_set να βρισκουμε οτι αυτη η τριαδα καρτων ειναι set. Αναλογα μαλιστα αν εχει ζητηθει απο τη κληση, δηλαδη αν εχουμε print =1, θα τυπωνει και ποιες ειναι αυτες οι καρτες. Ας περιγραψουμε ομως αναλυτικα

Γραμμη 125-126: Δηλωνουμε μεταβλητες w και z ως integer. Ακομη δηλωνουμε τη μεταβλητη set ως character και το θετουμε ισο με ‘0’.

Γραμμη 127-137: για δεσμευση της πρωτης καρτας χρησιμοποιουμε ενα διπλο for-loop που σκαναρει πρωτα τις σειρες απο i=0 ως i<3 και επειτα για καθε σειρα i σκαναρει τις στηλες απο απο j=0 ως j<4 . Και τα δυο loop τερματιζουν εαν το set γινει ισο με ‘1’

Γραμμη 129-137: για δεσμευση της δευτερης καρτας χρησιμοποιουμε ενα διπλο for-loop που σκαναρει πρωτα τις σειρες απο k=iως i<3 και επειτα για καθε σειρα k σκαναρει τις στηλες απο l=j ως l<4. Και τα δυο loop τερματιζουν εαν το set γινει ισο με ‘1’

Γραμμη 131-137: Επειδη πρεπει να εξασφαλισουμε οτι η δευτερη καρτα δεν ειναι ιδια με την πρωτη ελεγχουμε εαν ειτε η σειρα k ειτε η στηλη l ειναι διαφορετικες απο τη τρεχουσα πρωτη καρτα.

Γραμμη 132-137: για δεσμευση της τριτης καρτας χρησιμοποιουμε ενα διπλο for-loop που σκαναρει πρωτα τις σειρες απο w=k ως w<3 και επειτα για καθε σειρα w σκαναρει τις στηλες απο απο z=l ως z<4 . Και τα δυο loop τερματιζουν εαν το set γινει ισο με ‘1’

Γραμμη 134-137: εξασφαλιζουμε οτι η τριτη καρτα δεν ειναι ιδια με την δευτερη ελεγχοντας εαν ειτε η σειρα w ειτε η στηλη z ειναι διαφορετικες απο τη τρεχουσα δευτερη καρτα

Γραμμη 135-137: κανουμε το set ισο με το function test\_if\_set οπου βαζουμε ως συντεταγμενες τις συντεταγμενες των τρεχουσων τριων καρτων. Επειτα ελεγχουμε εαν το set = ‘1’ και print =’1’ (δηλαδη ο χρηστης ζητησε εμφανιση ποιες κανουν set) εμφανιζει τις συντεταγμενες τους

Γραμμη 138: κανουμε return το set.

Γραμμη 142-166: function kainourgio\_set(..) που θα χρησιμοποηθει για να κανουμε draw ενα συνολο καρτων. Δηλωνουμε πρωτα το set ως character και ισο με ‘0’. Επειτα κανουμε while-loop που επαναλαμβανεται μεχρι να γινει το set = ‘1’ (θα γινει ‘1’ οταν στον ελεγχο μεσω του cheat βρουμε οτι υπαρχει τουλαχιστον μια τριαδα που κανει σετ).

Γραμμη 146-162: ενα διπλο for-loop για οπου με το πρωτο σκαναρουμε τις σειρες απο i=0 ως i<3 και για καθε σειρα i σκαναρουμε τις στηλες απο j=0 ως j<4. Για καθε συνδιασμο i και j τραβαμε τυχαια μια karta[i][j] μεσω του function getNextCard και επειτα θετουμε το dublicate = 1 για να ξεκινησει το επομενο while-loop

Γραμμη 150-160: ενα while-loop που επαναλαμβανεται οσο το dublicate διαφορο του μηδενος.Επειτα θετουμε dublicate =0

Γραμμη 152-159: ενα διπλο for-loop οπου σκαναρει ολες τις προηγουμενες καρτες. Αυτο το διπλο loop τερματιζει εαν το dublicate γινει ισο με 1. Εαν καποια απο αυτες βγει ιδια μεσω strcmp των πεδιων enniaio τοτε τραβαμε κανουργια καρτα για τις συντεταγμενες i,j μεσω του karta[i][j] = getNextCard(..) και αλλαζουμε το dublicate =1 για να σταματησει το for-loop ενω το while-loop (γραμμη150) επαναλαμβανεται απο την αρχη. (προκειμενου να ελεγχθει αν η καινουργια καρτα ειναι ιδια με καποια προγενεστερη του τεστ)

Γραμμη 163: θετουμε set = cheat(..). Βαζουμε ως τελευταια μεταβλητη το μηδεν για να μην εμφανιση ποιες καρτες κανουν σετ. Εαν δεν βγει ισο με 1 επαναλαμβανεται το while-loop στη γραμμη 145. Ο λογος που το κανουμε αυτο ειναι για να εξασφαλισουμε οτι υπαρχει τουλαχιστον ενα σετ στο παιχνιδι. (περιττο ισως μιας και δεν το ζηταγε η ασκηση)

Γραμμη 169-326: το main του προγραμματος.

Γραμμη 170-174: Δηλωνουμε το δισδιαστατο πινακα karta[][] ως δομη KARTES. Επειτα δηλωνουμε τα i, j, k, l, dublicate, point και num ως integer οπου τα i, j, k και l θα χρησιμευσουν ως δεικτες/μετρητες, το dublicate για ελεγχο εαν εχουμε διπλη ιδια καρτα στο συνολο των καρτων, το point για την αξιολογηση αν εκανε set και το numως δεικτης του πινακα των παικτων. Επειτα δηλωνουμε τα prwth\_seira, prwth\_sthlh, deuterh\_seira, deuterh\_sthlh, trith\_seira και trith\_sthlh ως integer που θα χρησιμοποιησουμε για τις συντεταγμενες των τριων καρτων ενω δηλωνουμε και το epilogi[2] ως character για τις επιλογες εξοδου ή ενεργοποιηση παικτη. Τελος δηλωνουμε τους πινακες set = '1', set\_xrwmata[]= {'r', 'g', 'b'}, set\_sximata[]= {'c', 't', 'r'}, char set\_ufes[] = {'b', 'h', 'e'} και set\_arithmoi[] = { '1', '2', '3'} που θα χρησιμοποιησουμε κατα το τραβηγμα νεων καρτων.

Γραμμη 175-177: Δηλωνουμε το repeat ως character (αν τυχον γινει καποιο λαθος στη δηλωση ενεργου παικτη) και το πινακα paiktis[3] ως δομη PAIKTIS για τα στοιχεια των παικτων. Επειτα βαζουμε srand(time(NULL)) για αρχικοποιηση της μηχανης τυχαιων αριθμων.

Γραμμη 181-196: Γεμιζω το πινακα paiktis με τα στοιχεια των παικτων. Επειτα με ενα for-loop απο i = 0 εως i<3 αρχικοποιω τα σκορς των παικτων θετοντας paiktis[i].score=0. Αφου ολοκληρωθει με ενα for-loop εμφανιζω τα σκορς των παικτων. Τελος αρχικοποιω το epilogi[0] = ‘1’ για να ξεκινησει το επομενο while loop και epilog[1] = ‘\0’.

Γραμμη 197-323: while-loop που επαναλαμβανεται μεχρι να δωθει 0 για εξοδο

Γραμμη 201-202: Τραβα το καινουργιο συνολο καρτων μεσω του function kainourgio\_set. Για να γινει αυτο περναμε bye reference το karta και by value τα i, j, k, l, dublicate, set\_xrwmata, set\_sximata, set\_arithmoi, set\_ufes. Επειτα θετουμε set = ‘0’ για να ξεκινησει το επομενο while-loop.

Γραμμη 206-315: while-loop που επαναλαμβανεται μεχρι το set να γινει ισο με 1 . Εαν το set γινει ισο με 1 τερματιζει και ξεκιναμε απο το προηγουμενο while-loop οποτε στη γραμμη 201 τραβα καινουργιες καρτες.

Γραμμη 209-219: ξεκινα με ενα διπλο for-loop για την εμφανιση των καρτων. Επειτα εμφανιζει οδηγιες σχετικα με τις συντεταγμενες των καρτων και τελος θετουμε το repeat ισο με ‘1’ για να ξεκινησει το επομενο while-loop.

Γραμμη 220-241: θετει repeat ισο με ‘0’. Αν παραμεινει ως τελος τοτε ο χρηστης εδωσε ειτε εξοδο ειτε ενεργοποιησε σωστα καποιο παικτη. Ζηταμε απο το χρηστη να δωσει ειτε το γραμμα του id του παικτη που θα παιξει ειτε 0 για εξοδο και αποθηκευουμε την επιλογη του στο epilog[0]. Οποτε αν δωσει 0 με ενα for-loop εμφανιζει το σκορ των παικτων και τερματιζει με exit(1). Αν ομως δωσει a, g ή p θα θεσει το num ισο με 0, 1 ή 2 αντιστοιχα. Αν παλι δωσει C τοτε ενεργοποιει την κρυφη επιλογη cheat mode οπου θετει το repeat ισο με cheat(..)οπου εχουμε περασει τα karta, i, j, k, l και 1 προκειμενου να κανει printf μια τριαδα που κανουν σετ. Επειδη το cheat θα επιστρεψει τιμη ‘1’ τοτε και το repeat αλλαζει σε ‘1’ οποτε επαναλαμβανεται το while-loop. Τελος εαν δεν εδωσε κανενα απο τα προηγουμενα εμφανιζει μηνυμα λαθους και θετει repeat = ‘1’ για επαναλυψη.

Γραμμη 244-293: Ζητα απο τον ενεργο παικτη (εμφανιζει το ονομα του) να δωσει τις συντεταγμενες των 3 καρτων. Ζητα πρωτα την σειρα της πρωτης καρτας και το αποθηκευει στο prwth\_seira. Επειτα εχουμε while-loop που οσο η τιμη prwth\_seira που εδωσε ο χρηστης ειναι εκτος των οριων των σειρων (δηλαδη <0 ή >2) θα ζητα απο αυτον να δωσει εκ νεου τη σειρα. Ιδια διαδικασια επαναλαμβανεται για τη στηλη οπου αποθηκευεται στο prwth\_sthlh και γινεται while-loop μεχρι να δωσει ο χρηστης τιμη εντος των οριων των στηλων, δηλαδη να ειναι αναμεσα απο 0 εως 3. Τα παραπανω επαναλαμβανονται για τη δευτερη και τριτη καρτα οπου οι συντεταγμενες τους αποθηκευονται στα deuterh\_seira, deuterh\_sthlh, trith\_seira και trith\_sthlh αντιστοιχα.

Γραμμη 296-313: Πρωτα θετουμε set = test\_if\_set(..) οπου περναμε τις τιμες των karta, prwth\_seira, prwth\_sthlh, deuterh\_seira, deuterh\_sthlh, trith\_seira και trith\_sthlh. Εαν οι συντεταγμενες των 3 καρτων αποτελουν σετ τοτε θα επιστρεψει ‘1’ οποτε και το set γινεται ισο με ‘1’. Ετσι δεν θα επαναλυφθει το while-loop που ξεκινα στη γραμμη 206 αλλα θα ξεκινα απο γραμμη 197 (ετσι θα τραβηξει καινουργιες καρτες). Επειτα εαν το set = ‘1’ εμφανιζει σχετικο μηνυμα οτι εκανε σετ, αυξανει το πεδιο score του paikti[num] κατα 2 μοναδες εφοσον ειναι μικροτερο απο 254 (αλλιως το κανει ισο με 255) και το εμφανιζει. Εαν παλι το set δεν ειναι ‘1’ τοτε μειωνει το πεδιο score σκορ του paikti[num] κατα μια μοναδα εφοσον ειναι μεγαλυτερο του μηδενος (αλλιως το αφηνει ως εχει) και προχωρα σε εμφανιση του. Επειδη το set παρεμεινε ισο με ‘0’ ξεκινα παλι απο γραμμη 206 και επανεμφανιζει τις ιδιες καρτες.

Γραμμη 316-322: Εαν μετα την αλλαγη του σκορ εχουμε paiktis[num].score >=10 τοτε κερδισε την παρτιδα. Οποτε εμφανιζει σχετικο μηνυμα οτι κερδισε και με ενα for-loop προχωρα στην εμφανιση των σκορς ολων των παικτων. Τελος τερματιζει το προγραμμα με exit(1) (ενδεχομενως μπορουσε να τερματισει με αλλο τροπο, ελεγχοντας στο πρωτο while-loop το σκορ)