Θέμα 1

Στο θέμα αυτό χρησιμοποιήθηκαν με include τα <stdio.h> και <strdlib.h> τα οποία χρησιμοποιούνται σε κάθε προγραμμα σε C καθώς και το <string.h> για διαχείρηση αλφαριθμητικών, με εντολές όπως strcmp() κ.α.

Η κύρια δομή δεδομένων είναι η:

typedef struct {

unsigned char \*name;

unsigned short \*tel;

unsigned int seatNo;

} aBus;

η οποία χρησιμοποιήθηκε σε πίνακα 53 στοιχείων απο sizeof(aBus), όσος είναι και ο μέγιστος αριθμός διαθέσιμων θέσεων σε λεωφορείο. Με τον τρόπο αυτό γίνεται μια «σπατάλη μνήμης» όταν το κάθε λεωφορείο δεν είναι γεμάτο, κάτι που συμβαίνει πολλές φορές. Στη συνέχεια, για κάθε όνομα δεσμεύτηκε μνήμη 43 bytes για κάθε όνομα, και 11 bytes για κάθε τηλέφωνο.

Στη συνέχεια ανοίγετε το αρχείο «bus.txt» από όπου και διαβάζεται (εκτός απο τα τυποποιημένα αριθμό κυκλοφορίας και πλήθος του συγκεκριμένου λεωφορείου) τα στοιχεία κάθε δεσμευμένης θέσης, και μεταφέρονται στην αντίστοιχη προ-δεσμευμένη θέση του δυναμικού πίνακα.

Έπειτα υπάρχει μια δομή επανάληψης που διαχειρίζεται τον έλεγχο του προγράμματος για κατάλληλες διαδικασίες δίνοντας στο χρήστη ένα μενού επιλογών και περιγράφονται παρακάτω. Να τονισθεί ότι σε όλες τις λειτουργίες του μενού δεν γίνεται καμία δυναμική δέσμευση η απελευθέρωση μνήμης.

* Η εμφάνιση κενών θέσεων. Η λειτουργία επιτυγχάνεται με ένα απλό βρόχο επανάληψης που διατρέχει κάθε θέση του δυναμικού πίνακα και ελέγχοντας το πεδίο αριθμού θέσης, ελέγχει εάν η θέση είναι άδεια ή όχι.
* Η κράτηση συγκεκριμένης θέσης με αναφορά τον αριθμό της. Ελέγχεται εύκολα εάν είναι διαθέσιμη οπότε και καταχωρούνται τα στοιχεία που διαβάζονται.
* Εδώ γίνεται η κλασική αναζήτηση είτε με βάση το ονοματεπώνυμο είτε με βάση το τηλέφωνο του πιθανού επιβάτη, για να διαπιστωθεί έχει δεσμευθεί θέση σε αυτόν .
* Η ακύρωση θέσεως γίνεται και αυτή απλά με την εκχώρηση του ΜΗΔΕΝ στην αντίστοιχη θέση που δίνει ο χρήστης, εάν αυτή είναι κατειλημμένη.
* Με την τελευταία επιλογή εμφανίζεται λίστα με τα στοιχεία των πελατών, δηλαδή που έχουν ήδη δεσμεύσει θέση στον πίνακα.

Με την έξοδο από την επανάληψη το πρόγραμμα ανοίγει το αρχείο «bus.txt» για overwrite και γράφει τα νέα στοιχεία.

Σε όλη την διαδικασία δεν απαιτείται ταξινόμηση γιατί ο πίνακας χρησιμοποιείται με δείκτες (index και όχι pointers) σειριακού πίνακα.

Θέμα 2

Στο πρόγραμμα αυτό η δομή δεδομένων που κυριάρχησε ήταν η απλή συνδεδεμένη λίστα.

truct node {

unsigned char \*name;

unsigned short \*tel;

unsigned int seatNo

struct node \* next;

};

typedef struct node dynamicBus;

Τα πεδία δεδομένων είναι 2 δυναμικά (name, tel) και ένας ακέραιος (η αριθμημένη θέση λεωφορείου). Το πεδίο struct node \*next είναι ο δείκτης (pointer και όχι index) ο οποίος δείχνει στον επόμενο κόμβο.

Η διαχείριση της ουράς γίνεται με χρήση δύο κόμβων head και tail, οι οποίοι δείχνουν το πρώτο και το τελευταίο στοιχείο της λίστας, αντίστοιχα. Αρχικά δεν γίνεται καμιά δέσμευση μνήμης για καμιά θέση του λεωφορείου. Η διαχείριση είναι δυναμική και μνήμη δεσμεύεται μόνο κατά την νέα κράτηση θέσης και αποδεσμεύεται κατά την ακύρωση θέσης. Έτσι στο πρόγραμμα χρησιμοποιούνται διαδικασίες όπως εισαγωγή στην αρχή η στο τέλος μιας λίστας, εισαγωγή έπειτα από συγκεκριμένο κόμβο, διαγραφή συγκεκριμένου κόμβου, αναζήτηση κόμβου, ταξινόμηση με βάση κάποιο πεδίο δεδομένων (π.χ. name η seatNo).

Επίσης χρησιμοποιούνται συναρτήσεις όπως εμφάνιση λίστας, αντιγραφή λίστας, διαγραφή λίστας χρησιμοποιούνται είτε με αναδρομή είτε με δομή επανάληψης.

Σε περίπτωση εισαγωγής νέου κόμβου γίνεται και δέσμευση μνήμης (malloc) για τα πεδία του ονόματος και του τηλεφώνου, και φυσικά πριν κάθε διαγραφή τα ίδια πεδία επιστρέφουν την μνήμη που χρησιμοποιούσαν στο σύστημα με αποδέσμευση μνήμης (free).

Το πρόγραμμα κατά την εκκίνηση αρχικοποιεί δυο δείκτες (head, tail). Η διαχείρηση των πελατών/επιβατών γίνεται δυναμικά. Έτσι στην συνέχεια το πρόγραμμα διαβάζει το αρχείο «bus.txt» και διαβάζει τα στοιχεία των επιβατών κάνοντας κάθε φορά δημιουργία κόμβου στο τέλος της λίστας και εισάγοντας κατάλληλα τα στοιχεία τους (με αναδρομική διαδικασία). Αμέσως μετά γίνεται ταξινόμηση με βάση τον αριθμό θέσης του λεωφορείου. Αυτό γίνεται γιατί οι διαδικασίες που εκτελούνται παρακάτω έχουν ως προϋπόθεση ότι η λίστα είναι ταξινομημένη (π.χ. αναζήτηση με βάση τη θέση).

Η κεντρική δομή επανάληψης έχει τις κλασικές λειτουργίες της εισαγωγής σε συγκεκριμένη θέση, αναζήτηση με βάση κάποιο πεδίο δεδομένων, διαγραφή κόμβου και τέλος ταξινόμηση με βάση κάποιο πεδίο του κόμβου. Η ταξινόμηση που χρησιμοποιήθηκε είναι η ταξινόμηση με επιλογή (selection sort) κατά την οποία επαναληπτικά από κάποιο στοιχείο i, μέχρι το τέλος της λίστας, βρίσκουμε το ελάχιστο και το ανταλλάσσουμε με το τρέχων στοιχείο i.

Μετά το τέλος της κεντρικής επανάληψης η οποία διαχειρίζεται το μενού επιλογών, τα στοιχεία αντιγράφονται αναδρομικά σε μια νέα λίστα (με προσθήκη στοιχείων στο τέλος της). Στη συνέχεια με άλλη αναδρομική διαδικασία, αποθηκεύονται τα δεδομένα κάθε κόμβου στο ίδιο αρχείο «bus.txt» και κατόπιν ο κόμβος διαγράφεται.

Πρέπει να αναφερθεί ότι χρησιμοποιήθηκαν και κάποιες βοηθητικές δυναμικές μεταβλητές με ολική δήλωση τους (global), για να γίνει μόνο μια φορά (στην αρχή), η δέσμευση μνήμης και μόνο μια φορά (στο τέλος) η αποδέσμευσή τους και να είναι διαθέσιμες σε όλες τις συναρτήσεις/διαδικασίες.Θέμα 3

Αρχικά θα περιγράφουνε οι δομές δεδομένων.

Η δομή για κάρτες περιέχει ένα χαρακτήρα για κάθε μία από τις ιδιότητες χρώμα, σχήμα, αριθμό και υφή. Στην συνέχεια χρησιμοποιείται πίνακας 4 διαστάσεων και συνολικού πλήθους 81=34 από την παραπάνω δομή με το όνομα mycards[3][3][3][3] και αρχικοποιείται με όλους τους πιθανούς συνδυασμούς ώστε να έχουμε τον πίνακα με τις 81 κάρτες..

typedef struct {

char xrwma,

sxhma,

arithmos,

yfh;

} SetCards;

SetCards myCards[3][3][3][3];

Μετά την εμφάνιση των 12 καρτών σε πλέγμα 3x4, ο χρήστης καλείται να επιλέξει τρείς από τις 12 κάρτες με χρήση συντεταγμένων χ και ψ. Η πρώτη τιμή για το χ έχει πιθανές τιμές 0, 1 και 2 ενώ το ψ έχει πιθανές τιμές 0,1,2 και 3. Για κάθε ζεύγος χρησιμοποιείται η δομή Points ενώ για τα τρία ζεύγη χρησιμοποιείται ο πίνακας myPoints[3] από Points.

typedef struct {

int x,y; /\*(γραμμή 0 έως 2) - (στήλη 0 έως 3) \*/

} Points;

Points myPoints[3];

ΠΡΟΣΒΑΣΗ ΣΤΙΣ ΚΑΡΤΕΣ ΚΑΙ ΤΙΣ ΙΔΙΟΤΗΤΕΣ ΤΟΥΣ

Για τις δώδεκα κάρτες που επιλέγονται αποθηκεύουμε τους 12 αντίστοιχους δείκτες (με τιμές στο εύρος 0 έως 80) σε πίνακα ακεραίων dwdeka[12]. Αυτό σημαίνει ότι κάθε i κάρτα από τις 12, θα είναι η deikths = dwdeka[4\*myPoints[k].x + myPoints[k].y] (όπου κ=0,1 και 2). Ο δείκτης αυτός θα έχει τιμή από το 0 μέχρι το 80. Για να εντοπίσουμε την κάρτα και τις τιμές των ιδιοτήτων χρησιμοποιούμε ένα pointer p, απο SetCards \* και έτσι οι τρεις κάρτες που έχουν επιλεγεί είναι οι p[deikths], για k = 0,1,2.

Για την επιλογή 12 τυχαίων καρτών χρησιμοποιείται η srand(time(NULL)) για να επιλέγονται κάθε φορά νέοι τυχαίοι αριθμοί καθώς και η rand() για να επιλέγονται αριθμοί στο διάστημα που επιθυμούμε (συνήθως 0 – 80).

Όσον αφορά τους παίκτες, η αντίστοιχη δομή περιλαμβάνει ένα χαρακτήρα για id, το όνομα 17 χαρακτήρων και το score. Στη συνέχεια για τους παίκτες χρησιμοποιείται ο πίνακας setPaikths[3] (το λεκτικό PAIKTES είναι ορισμένο ίσο με 3 με #define) στον preprocessor.

typedef struct {

char id;

int score;

char name[17];

} SetPaiktes;

SetPaiktes setPaikths[PAIKTES];

ΤΑ ΒΑΣΙΚΑ ΣΗΜΕΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Το κυρίως πρόγραμμα ξεκινάει με αρχικοποιήσεις για τις κάρτες και τους παίκτες. Ανακατεύει τις κάρτες και επιλέγει τις πρώτες 12 κάρτες και ζητάει από τους χρήστες να επιλέξουν ποιος θα παίξει πληκτρολογώντας το id τους. Στην συνέχεια χρησιμοποιεί δομή επανάληψης μέχρι να πληκτρολογηθεί το ‘0’ ή μέχρι να φτάσει κάποιος παίκτης τους 10 πόντους.

Η μέγιστη βαθμολογία αρχικοποιείται με 0 και έπειτα από κάθε ΣΕΤ που ανακαλύπτει κάποιος χρήστης τότε ελέγχεται η αυξημένη βαθμολογία του χρήστη, για να βρεθεί μήπως υπάρχει νέος μέγιστος.

Η συνάρτηση getNextCard() δέχεται για παραμέτρους τον πίνακα με τους δείκτες των 12 καρτών καθώς και το πλήθος όσων έχουν ήδη επιλεγεί. Τότε με έναν επαναληπτικό βρόχο

Επιλέγει τυχαίες κάρτες μέχρι να βρει κάποια η οποία δεν υπάρχει στον πίνακα. Ο έλεγχος αυτός γίνεται με αναζήτηση της τυχαία επιλεγόμενης κάρτας στα ήδη υπάρχοντα στοιχεία του πίνακα με δείκτες από 0 μέχρι και ένα λιγότερο από τα ήδη επιλεγμένα.

Η συνάρτηση αυτή καλείται 11 φορές καθώς το πρώτο στοιχείο απλά είναι κάποιο τυχαίο από τα 81.

Τέλος πρέπει να αναφερθεί ότι έχει ορισθεί στην αρχή του προγράμματος με #define MYTEST, το οποίο όταν είναι ενεργό (χωρίς σχόλια) τότε αμέσως μετά την εμφάνιση του πλέγματος εμφανίζει και όλες τις τριάδες οι οποίες δημιουργούν SET. Αυτό υλοποιήθηκε για να ελεγχθεί το πρόγραμμα από πλευράς επιλογής και εμφάνισης του πλέγματος και της συνάρτησης που ελέγχει εάν αποτελεί SET μία τριάδα.