Εργασία 1 – Βάση δεδομένων στη μνήμη με δυναμικό πίνακα

Αναπτύξτε ένα πρόγραμμα για την διαχείριση του μητρώου των φοιτητών του τμήματος, όπου όλα τα δεδομένα βρίσκονται στην μνήμη του υπολογιστή. Η επιθυμητή λειτουργικότητα περιγράφεται παρακάτω. Το πρόγραμμα που θα υποβάλετε πρέπει να είναι αποθηκευμένο σε ένα μόνο αρχείο με όνομα `project1.c`

Εγγραφές

Κάθε εγγραφή περιέχει την εξής πληροφορία: (1) ΑΕΜ -- long unsigned int, (2) όνομα -- char[64], (3) αριθμός χρωστούμενων μαθημάτων -- short unsigned int.

Το πρόγραμμα πρέπει να χρησιμοποιεί μια κατάλληλη δομή (struct) για την αποθήκευση αυτών των δεδομένων. Κάθε εγγραφή πρέπει να αποθηκεύεται ξεχωριστά, σε μνήμη που δεσμεύεται δυναμικά.

Παράλληλα, το πρόγραμμα πρέπει να διατηρεί έναν πίνακα με δείκτες στις εγγραφές (πίνακας δεικτών). Το μέγεθος του πίνακα πρέπει να αυξομειώνεται δυναμικά, ανάλογα με τον αριθμό των εγγραφών, αυξάνοντας το μέγεθος του πίνακα κατά Κ εγγραφές όταν δεν υπάρχει χώρος για την προσθήκη νέας εγγραφής και μειώνοντας το μέγεθος του κατά Κ εγγραφές όταν προκύπτουν περισσότερες από Κ κενές θέσεις μετά τη διαγραφή μιας εγγραφής, όπου Κ μια τιμή που το πρόγραμμα δέχεται ως όρισμα.

Λειτουργίες

Το πρόγραμμα παίρνει ως ορίσματα από τη γραμμή εντολών το αρχικό μέγεθος Μ του πίνακα δεικτών, και το μέγεθος Κ κατά το οποίο αυξομειώνεται ο πίνακας όπως περιγράφεται παραπάνω. Πρέπει να υποστηρίζονται οι εξής λειτουργίες (η ακριβής περιγραφή των εντολών που πρέπει να μπορεί να διαβάσει από την είσοδο του το πρόγραμμα και των απαντήσεων που πρέπει να εκτυπώνει στην έξοδο του, δίνεται σε ξεχωριστή ενότητα):

add: Προσθήκη εγγραφής, με δέσμευση μνήμης για την αποθήκευση των δεδομένων της εγγραφής, και προσθήκη δείκτη στον πίνακα δεικτών (με κατάλληλη αύξηση του μεγέθους του πίνακα). Αν υπάρχει εγγραφή με το ίδιο ΑΕΜ, η προσθήκη αποτυγχάνει.

rmv: Απομάκρυνση εγγραφής με βάση το ΑΕΜ, με απελευθέρωση της μνήμης της εγγραφής, και αφαίρεση του αντίστοιχου δείκτη από τον πίνακα δεικτών (με κατάλληλη μείωση του μεγέθους του πίνακα δεικτών). Αν δεν υπάρχει εγγραφή με το δεδομένο ΑΕΜ, η απομάκρυνση αποτυγχάνει.

mod: Τροποποίηση εγγραφής (αλλαγή του αριθμού χρωστούμενων μαθημάτων) με βάση το ΑΕΜ. Αν δεν υπάρχει εγγραφή με το δεδομένο ΑΕΜ, η τροποποίηση αποτυγχάνει.

find: Αναζήτηση εγγραφής με βάση το ΑΕΜ.

sort: Ταξινόμηση του πίνακα δεικτών, με αύξουσα σειρά ΑΕΜ των αντίστοιχων εγγραφών.

print: Εκτύπωση όλων των εγγραφών.

clear: Διαγραφή όλων των εγγραφών με αντίστοιχη απελευθέρωση της μνήμης (με αντίστοιχη μείωση του μεγέθους του πίνακα δεικτών στο μηδέν).

quit: Τερματισμός του προγράμματος, με διαγραφή όλων των εγγραφών (βλέπε άνω).

Απαιτήσεις υλοποίησης

- Απαγορεύεται η χρήση καθολικών (global) ή static μεταβλητών και goto.
- Τα ονόματα των φοιτητών πρέπει να αποθηκεύονται στις δομές με κεφαλαία. Υποθέστε ότι τα ονόματα που δέχεται το πρόγραμμα από την είσοδο του δεν έχουν τόνους ούτε περιέχουν κενά, μπορεί όμως να μην είναι με κεφαλαία γράμματα οπότε το πρόγραμμα πρέπει να τα μετατρέπει σε κεφαλαία προτού προχωρήσει σε αποθήκευση ή αναζήτηση.
- Κάθε μια από τις παραπάνω λειτουργίες πρέπει να υλοποιηθεί ως ξεχωριστή συνάρτηση, που θα καλείται μέσα από την main του προγράμματος ανάλογα με τις εντολές που δέχεται το πρόγραμμα.
- Η ταξινόμηση του πίνακα δεικτών πρέπει να υλοποιηθεί με τον αλγόριθμο insertion sort.
- Εάν ο πίνακας είναι πλήρως ταξινομημένος, η αναζήτηση πρέπει να εκμεταλλευτεί αυτό το γεγονός και να υλοποιηθεί με τον αλγόριθμο δυαδικής αναζήτησης (binary search), διαφορετικά θα γίνεται με γραμμική αναζήτηση (linear search).
- Η βασική λειτουργία της αναζήτησης με βάση το ΑΕΜ πρέπει να υλοποιηθεί μια μοναδική φορά, έτσι ώστε να μπορεί να χρησιμοποιείται στις λειτουργίες add, rmv, mod, find.
- Για κάθε κλήση των λειτουργιών sort και find πρέπει να μετριέται και να εκτυπώνεται στην έξοδο σφαλμάτων (stderr) ο αριθμός των συγκρίσεων που πραγματοποιούνται.
- Η προσθήκη νέας εγγραφής γίνεται πάντα στην πρώτη κενή θέση του πίνακα δεικτών.
- Μετά την ταξινόμηση πρέπει όλες οι κενές θέσεις του πίνακα δεικτών, αν υπάρχουν, να βρίσκονται στο τέλος (να μην υπάρχουν ενδιάμεσα κενά).

Μελέτη απόδοσης

Μελετήστε την απόδοση της υλοποίησης σας, για τυπικά σενάρια εκτέλεσης. Συγκεκριμένα:

- Καταγράψτε τον αριθμό των συγκρίσεων της εντολής sort, την πρώτη φορά που καλείται, για πίνακα με 1.000, 10.000 και 100.000 εγγραφές.
- Καταγράψτε τον αριθμό των συγκρίσεων της εντολής find, για ΑΕΜ που υπάρχουν και ΑΕΜ που δεν υπάρχουν, όταν ο πίνακας δεν είναι πλήρως ταξινομημένος και έχει 1.000, 10.000 και 100.000 εγγραφές.
- Καταγράψτε τον αριθμό των συγκρίσεων της εντολής find, για ΑΕΜ που υπάρχουν και ΑΕΜ που δεν υπάρχουν, όταν ο πίνακας είναι πλήρως ταξινομημένος, και έχει 1.000, 10.000 και 100.000 εγγραφές.

Για κάθε μια μέτρηση, θα βρείτε στο site του μαθήματος ένα text file που σε πρώτη φάση περιέχει τις εντολές προσθήκης εγγραφών (για να γεμίσει η βάση), και σε δεύτερη φάση την εντολή που θέλουμε να μετρήσουμε (μια ή περισσότερες φορές, αναλόγως με την μέτρηση). Εσείς απλά ανακατευθύνετε την είσοδο του προγράμματος στο αρχείο, και την έξοδο σφαλμάτων σε ένα αρχείο που θα αποθηκευτούν όλες οι μετρήσεις. Όταν τερματίσει η εκτέλεση, εξάγετε από το αρχείο τις τιμές που εκτύπωσε το πρόγραμμα, και τις επεξεργάζεστε, π.χ., με την βοήθεια του Excel, για να βγάλετε τα βασικά στατιστικά μεγέθη (min, max, average, median).

Τέλος, θα πρέπει να φτιάξετε γραφικά διαγράμματα (graph plots) με τα παραπάνω αποτελέσματα, τα οποία θα παρουσιάσετε/σχολιάσετε κατά την προφορική εξέταση της εργασίας.

Μορφοποίηση εισόδου/εξόδου

Παρακάτω ορίζεται η ακριβής μορφή των εντολών που διαβάζει το πρόγραμμα από την είσοδο του, καθώς και η μορφή των απαντήσεων που εκτυπώνει το πρόγραμμα στην έξοδο του και στην έξοδο σφαλμάτων.

Εντολή	Είσοδος	Έξοδος stdout (επιτυχία)	Έξοδος stdout (αποτυχία)	Έξοδος stderr
add	a <aem> <name> <courses>	A-OK <aem>, <e> <c>	A-NOK <aem>, <e> <c>	-
rmv	r <aem>	R-OK <aem>, <e> <c>	R-NOK <aem>, <e> <c>	-
mod	m <aem> <courses>	M-OK <aem>	M-NOK <aem>	-
sort	s	S-OK	-	$<nof comps>
find	f <aem>	F-OK <name> <courses>	F-NOK <aem>	$<nof comps>
print	p	* βλέπε παρακάτω	-	-
clear	c	C-OK	-	-
quit	q	-	-	-

Το <e> είναι το πλήθος εγγραφών και το <c> είναι το μέγεθος του πίνακα δεικτών.

Αν μια εντολή αποτελείται από περισσότερα τμήματα (συνοδεύεται από μια ή περισσότερες παραμέτρους), αυτά διαχωρίζονται με έναν ή περισσότερους χαρακτήρες ‘ ‘ (space). Κάθε εντολή τερματίζει με τον χαρακτήρα ‘\n’.

Πριν και μετά από κάθε μήνυμα εξόδου εκτός της εντολής print βρίσκεται χαρακτήρας ‘\n’. Αν το μήνυμα αποτελείται από περισσότερα τμήματα, αυτά διαχωρίζονται από έναν μοναδικό χαρακτήρα ‘ ‘ (space) .

* Στην εντολή print, εκτυπώνονται αρχικά οι χαρακτήρες '\n', ‘#’, ‘\n’ και μετά μία-μία οι εγγραφές με χαρακτήρα ‘\n’ στο τέλος κάθε εγγραφής. Τα πεδία μιας εγγραφής εμφανίζονται με τη σειρά <aem> <name> <courses> με έναν χαρακτήρα ‘ ‘ (space) ανάμεσά τους.

Εκτύπωση στο stderr γίνεται με τη συνάρτηση fprintf ως εξής:

```
fprintf(stderr, "\n$%d\n", comparisons);
```

όπου comparisons ακέραια μεταβλητή στην οποία αποθηκεύεται το πλήθος συγκρίσεων.

Προσοχή: Το πρόγραμμα πρέπει να ακολουθεί ακριβώς τις παραπάνω προδιαγραφές, διαφορετικά θα αποτυγχάνει ο αυτοματοποιημένος έλεγχος της λειτουργίας του προγράμματος, όπου η είσοδος και η έξοδος του προγράμματος ανακατευθύνεται σε αρχεία, όπου τα αρχεία εισόδου περιέχουν τις εντολές ακριβώς με βάση την παραπάνω μορφή, ενώ τα αρχεία εξόδου συγκρίνονται, byte προς byte, με άλλα αρχεία που έχουν ακριβώς την αναμενόμενη έξοδο. Επίσης, δεν θα λειτουργήσουν σωστά τα αρχεία εισόδου για τις μετρήσεις που πρέπει να κάνετε (βλέπε παραπάνω).

Για όσους έχουν κέφι

- Επεκτείνετε την υλοποίηση σας έτσι ώστε, εκτός από τον αριθμό συγκρίσεων, να τυπώνεται στο stderr και ο χρόνος που χρειάστηκε η αντίστοιχη λειτουργία (ταξινόμηση / αναζήτηση)
- Προσθέστε ως εναλλακτικό αλγόριθμο ταξινόμησης του πίνακα δεικτών, τον quicksort. Επαναλάβετε τις μετρήσεις, παρατηρήστε τις διαφορές στην απόδοση. Για αποφυγή stack smashing για μεγάλα βάθη αναδρομής, μπορείτε να κρατάτε την κατάσταση της αναδρομής σε δυναμική μνήμη που θα διαχειρίζεστε εσείς.

Σημαντικές ημερομηνίες

~~*Πέμπτη 20/2/2020*: Φροντιστήριο (στην ώρα/αίθουσα του μαθήματος)~~

Κυριακή 8/3/2020, 21:00 : Προθεσμία υποβολής

Τετάρτη ή/και Πέμπτη 11-12/3/2020 : Εξέταση εργασιών στο εργαστήριο (θα αναρτηθούν ώρες αργότερα).

Οδηγίες αποστολής εργασιών

Η αποστολή των εργασιών θα γίνεται μέσω autolab, στο μάθημα ECE116-S20. Απαιτείται σύνδεση στο VPN του πανεπιστημίου.

Από τη στιγμή που θα κάνετε την πρώτη υποβολή εργασίας στο autolab, δεσμεύεστε ότι θα ασχοληθείτε με τις εργασίες σε αυτό το εξάμηνο.

Συνιστούμε να κάνετε πρώτα έλεγχο ορθότητας του προγράμματός σας τοπικά στο PC σας χρησιμοποιώντας τα αρχεία ελέγχου που θα σας δώσουμε και όταν είστε σίγουροι ότι λειτουργεί, να κάνετε υποβολή στο autolab.

Πρόσβαση στο autolab

Όσοι δηλώσατε συμμετοχή στις εργασίες, έχετε ήδη εγγραφεί στο μάθημα ECE116-S20 του autolab με username το email της σχολής με κατάληξη @uth.gr

Αν δε θυμάστε τον κωδικό σας από το προηγούμενο εξάμηνο, ή δεν είχατε λογαριασμό με username αυτής της μορφής, τότε κάντε click στο "Forgot your password?", εισάγετε το email της σχολής με κατάληξη @uth.gr και πατήστε "Send me reset password instructions". Ελέγξτε το email σας για ένα μήνυμα με αποστολέα το ce120lab@gmail.com που θα περιέχει ένα link μέσω του οποίου μπορείτε να κάνετε reset τον κωδικό σας.

Πακετάρισμα και αποστολή εργασίας

1. Αποθηκεύστε το πρόγραμμά σας σε αρχείο με όνομα `project1.c`
2. Κατασκευάστε ένα κατάλογο με όνομα `project1submit`
3. Αντιγράψτε το `project1.c` στον κατάλογο `project1submit`

4. Πακετάρετε και συμπιέστε τον κατάλογο `project1submit` κάνοντας δεξί κλικ κι επιλέγοντας `Compress here as .tar.gz`

5. Κάντε login στο autolab, κι επιλέξτε το `project1` του μαθήματος ECE116-S20

6. Αποδεχτείτε το μήνυμα ακαδημαϊκής ακεραιότητας και μετά κάντε κλικ στο `SUBMIT`.

7. Στο παράθυρο που θα εμφανιστεί εντοπίστε κι επιλέξτε το αρχείο `project1submit.tar.gz` που κατασκευάσατε ώστε να το ανεβάσετε στο autolab.

8. Μετά από ένα λεπτό, ανανεώσετε τη σελίδα για να δείτε το βαθμό σας στα επιμέρους τεστ. Θα πρέπει να έχετε 100 μονάδες.