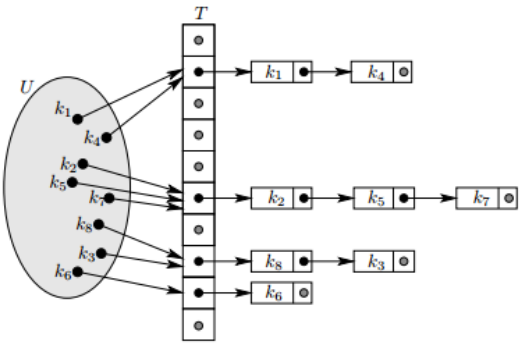
Εργαστήριο Δομών Δεδομένων - 3η Άσκηση
Ημερομηνία Παράδοσης : 07/12/2016 (πριν την έναρξη του εργαστηρίου - 11.00 -)

Εργαστηριακός Διδάσκων Μαθήματος: Δούμα Αναστασία (sia@aegean.gr)

Στη τρίτη εργαστηριακή άσκηση καλείστε να μελετήσετε τη χρήση των πινάκων κατακερματισμού (hash tables) και να αναπτύξετε σχετική εφαρμογή. Στα πλαίσια της 3ης άσκησης θα ασχοληθείτε και με την δομή δεδομένων του σωρού.

Ζήτημα 1 (Βαρύτητα 50%) - Κατακερματισμός

Σε πολλές εφαρμογές απαιτείται η χρήση δομών δεδομένων όπου οι ενέργειες της εισαγωγής, αναζήτησης και διαγραφής να εκτελούνται πολύ αποδοτικά. Σε τέτοιες περιπτώσεις μια λύση είναι ο κατακερματισμός. Ένας πίνακας κατακερματισμού (Hash table) είναι μία αποδοτική δομή δεδομένων για υλοποίηση λεξικών. Ο αναμενόμενος μέσος χρόνος εισαγωγής, αναζήτησης και διαγραφής ενός στοιχείου από έναν πίνακα κατακερματισμού είναι O(1).



Η βασική λειτουργία του κατακερματισμού είναι η εξής: Κάθε στοιχείο με κλειδί *k* αποθηκεύεται στη θέση *h(k)* του πίνακα *T*. Η συνάρτηση *h* καλείται συνάρτηση κατακερματισμού (***hash function***) και όπως φαίνεται και από τον πίνακα απεικονίζει τα k κλειδιά σε ένα πίνακα *m* θέσεων. Η συνάρτηση κατακερματισμού ανεξάρτητα από το τύπο και την τιμή του κλειδιού θα παράγει μια ακέραια τιμή στο εύρος από 0 έως m-1.

Μια συχνά χρησιμοποιούμενη συνάρτηση κατακερματισμού h είναι η ακόλουθη : *h(k) = k mod m.* Το m πρέπει να πληροί κάποιες προϋποθέσεις όπως να είναι πρώτος αριθμός. Αν τα κλειδιά είναι αρνητικοί αριθμοί ή συμβολοσειρές πρώτα μετατρέπονται σε μη αρνητικούς ακεραίους.

Ιδανικά θα επιθυμούσαμε διαφορετικά κλειδιά να απεικονίζονται σε διαφορετικές διευθύνσεις. Υπάρχει βέβαια πρόβλημα όταν δύο διαφορετικά κλειδιά απεικονίζονται στην ίδια θέση του πίνακα. Αυτή η κατάσταση λέγεται **σύγκρουση** (***collision***). Η συνάρτηση κατακερματισμού θα πρέπει να είναι έτσι φτιαγμένη ώστε να παράγει τέτοιες αντιστοιχίσεις που να ελαχιστοποιούνται οι συγκρούσεις. Ωστόσο, όταν το σύνολο των κλειδιών είναι μεγαλύτερο από την διάσταση m του πίνακα, θα υπάρχουν σίγουρα συγκρούσεις. Για την επίλυση τους χρησιμοποιούνται αποδοτικοί αλγόριθμοι, όπως αυτός της επίλυσης των συγκρούσεων με λίστες αλλά και της ανοιχτής διευθυνσιοδότησης. Στην παρούσα άσκηση θα ασχοληθούμε με τον πρώτο τρόπο.

Πιο συγκεκριμένα στην άσκηση αυτή θα υλοποιήσετε μία εφαρμογή για αυθεντικοποίηση χρηστών χρησιμοποιώντας πίνακα κατακερματισμού. Ένας πίνακας κατακερματισμού θα

μπορούσε να ήταν χρήσιμος για την αποθήκευση των ονομάτων χρηστών (login names) και των συνθηματικών.

Κατά την εκκίνηση της εφαρμογής σας το πρόγραμμα θα πρέπει να δημιουργεί δυναμικά έναν σταθερού μεγέθους πίνακα κατακερματισμού. Επιλέξτε εσείς πιο θα είναι το μέγεθος του. Ακολούθως θα διαβάζει ένα αρχείο κειμένου με όνομα «passwords.dat» το οποίο θα περιέχει ανά γραμμή το όνομα και το συνθηματικό κάθε χρήστη που το σύστημα μας αυθεντικοποιεί. Δηλαδή η μορφή του αρχείου θα μπορούσε να είναι :

```
john #12345!
george qwerty@
anna !AP2016!
mairi 9876m
```

Κάθε φορά που διαβάζεται το ζευγάρι των πληροφοριών «όνομα χρήστη» και «συνθηματικό» για κάθε χρήστη θα πρέπει να προστίθενται στον πίνακα κατακερματισμού χρησιμοποιώντας ως κλειδί *k* το όνομα του χρήστη.

Για την μετατροπή του κλειδιού από συμβολοσειρά σε ακέραιο μπορείτε να παράγετε ένα άθροισμα από την asci τιμή που αντιστοιχεί σε κάθε χαρακτήρα. Εναλλακτικά θα μπορούσατε να χρησιμοποιήσετε εσείς έναν αλγόριθμο επιλογής σας. Για hash συνάρτηση χρησιμοποιήστε την $h(k) = k \bmod m$ όπου m το μέγεθος του πίνακα κατακερματισμού. Σε περιπτώσεις συγκρούσεων θα πρέπει να χρησιμοποιήσετε μονοδεσμικές λίστες. Κάθε νέο κλειδί που προστίθεται σε λίστα λόγω σύγκρουσης με προηγούμενα κλειδιά θα πρέπει να προστίθεται στην αρχή της.

Από την στιγμή που ολοκληρωθεί η ανάγνωση του αρχείου και η αποθήκευση των πληροφοριών των χρηστών η εφαρμογή θα πρέπει να παρέχει τη δυνατότητα της αυθεντικοποίησης των χρηστών. Πιο συγκεκριμένα, θα ζητάει το όνομα χρήστη και το συνθηματικό για τον οποίο θέλουμε να γίνει η αυθεντικοποίηση. Ακολούθως θα αναζητά το συνθηματικό που είναι αποθηκευμένο στον πίνακα κατεκερματισμού για αυτόν τον χρήστη και εφόσον το βρει θα συγκρίνει το αποθηκευμένο συνθηματικό με αυτό που έχει εισάγει ο χρήστης. Αν είναι το ίδιο θα ολοκληρώνεται η διαδικασία με επιτυχία στην διαδικασία της αυθεντικοποίησης. Σε περίπτωση που δεν υπάρχει ο χρήστης θα πρέπει να εμφανίζεται σχετικό μήνυμα. Τέλος ο χρήστης θα πρέπει να ερωτάται αν θέλει να επαναλάβει τη λειτουργία της αυθεντικοποίησης.

Ενδεικτικός διάλογος από την εκτέλεση του προγράμματος θα μπορούσε να είναι ο ακόλουθος:

```
Enter Login Name : george
Enter Password : 123456

Authentication Fail

Do you want to continue (y/n) ? y


Enter Login Name : george
Enter Password : qwerty@

Authentication Successful

Do you want to continue (y/n) ? y


Enter Login Name : bill
Enter Password : qazwsx
```

```
Unknown User Name
Do you want to continue (y/n) ? n
```

Όλες οι βασικές λειτουργίες της εφαρμογής θα πρέπει να υλοποιηθούν με χρήση συναρτήσεων. Συναρτήσεις που θα πρέπει **απαραιτήτως** να υλοποιηθούν είναι οι :

`InitializeHashTable()`: Δημιουργεί και αρχικοποιεί κατάλληλα τον πίνακα κατεκερματισμού

`GetKey()`: επιστρέφει μια ακέραια τιμή που αντιστοιχεί σε συμβολοσειρά που δέχεται ως παράμετρο

`Hash()`: Υπολογίζει τη θέση του πίνακα που αντιστοιχεί στο ακέραιο κλειδί key χρησιμοποιώντας τη συνάρτηση κατακερματισμού που αναφέρθηκε.

`insert()`: Εισάγει τις πληροφορίες αυθεντικοποίησης ενός χρήστη στη δομή κατακερματισμού. Πρώτα υπολογίζει ένα ακέραιο κλειδί με κλήση της getKey() και στη συνέχεια υπολογίζει τη θέση που αντιστοιχεί στο κλειδί με κλήση της hash().

`search()`: Αναζητά το συνθηματικό ενός χρήστη στη δομή.

`print()`: Εμφανίζει για κάθε θέση του πίνακα κατακερματισμού τα δεδομένα που έχουν αποθηκευτεί. Σκόπιμο είναι να καλέσετε αυτή τη συνάρτηση για να εμφανίσετε τον πίνακα όπως έχει διαμορφωθεί ακριβώς μετά την ανάγνωση του αρχείου «passwords.dat».

Το ποια θα είναι η υπογραφή της κάθε συνάρτησης το αποφασίζετε εσείς ανάλογα με την υλοποίηση σας.

Ζήτημα 2 (Βαρύτητα 50%) – Ουρά προτεραιότητας

Υλοποιήστε πρόγραμμα το οποίο θα διαβάζει από έναν μονοδιάστατο πίνακα 20 στοιχείων τους τυχαίους αριθμούς που εμπεριέχει και στη συνέχεια θα τους αποθηκεύει σε ένα σωρό μεγίστου (δεύτερος πίνακας – δομή σωρού). Οι τυχαίοι αριθμοί στον πίνακα θα πρέπει να παίρνουν τιμές από το 1 έως 100.

Η διαχείριση του σωρού θα γίνεται μέσω του παρακάτω μενού επιλογών:

1. Δημιουργία πίνακα τυχαίων
2. Δημιουργία σωρού από πίνακα
3. Προσθήκη νέου στοιχείου στο σωρό (από χρήστη)
4. Εμφάνιση πίνακα σωρού
5. Εμφάνιση σωρού σε «δενδροειδή» αναπαράσταση (απλή εκτύπωση τιμών ανά επίπεδο δέντρου σωρού)
6. Ταξινόμηση κι εμφάνιση ταξινομημένου πίνακα
7. Έξοδος

Η επιλογή «*Ταξινόμηση κι εμφάνιση ταξινομημένου πίνακα*» θα δέχεται ως είσοδο τον πίνακα των τυχαίων αριθμών από την 1η επιλογή και θα έχει ως έξοδο τον ταξινομημένο πίνακα. Η ταξινόμηση θα γίνεται με χρήση σωρού (HeapSort).

Όλες οι βασικές λειτουργίες της εφαρμογής θα πρέπει να υλοποιηθούν με χρήση συναρτήσεων.

Παραδοτέα:
α) Τον κώδικα που θα έχετε υλοποιήσει μέσα στο εργαστηρίου θα πρέπει να τον παραδώσετε με την ολοκλήρωση του εργαστηρίου (ισχύει μόνο για όσους παρακολουθούν το εργαστήριο). Υπάρχει σχετικός σύνδεσμος στο eclass. Το αρχείο που θα ανεβάσετε θα πρέπει να έχει όνομα ***Exercise3_lab.cpp (ή .c - θα πρέπει να το κάνετε .zip για να το ανεβάσετε). Στην 1η γραμμή του κώδικα σας θα πρέπει υποχρεωτικά να αναγράφεται το όνομα και ο αριθμός μητρώου των μελών της ομάδας που συμμετείχε στην υλοποίηση που έγινε στο εργαστήριο.***

(β) Τελική παράδοση εργασίας :
Για την τελική παράδοση της εργασίας θα πρέπει να δημιουργήσετε **δύο αρχεία** με τον πηγαίο κώδικα για κάθε ένα από τα ζητήματα της άσκησης και όνομα ***Exercise31.cpp (ή .c)*** *και* ***Exercise32.cpp (ή .c).***

Τέλος θα δημιουργήσετε ένα zip αρχείο με όνομα αρχείου «το email σας_όνομα άσκησης» (πχ. icsd000666_Exercise3.zip) (μόνο ένα μέλος της ομάδας θα πρέπει να καταγράψει το email του). Το zip αρχείο θα πρέπει να περιέχει τα παραπάνω δύο αρχεία. ΠΡΟΣΟΧΗ: ΚΑΜΙΑ ΕΡΓΑΣΙΑ ΔΕΝ ΘΑ ΔΙΟΡΘΩΘΕΙ ΑΝ ΔΕΝ ΑΠΟΣΤΑΛΕΙ ΜΕ ΤΟΝ ΠΑΡΑΠΑΝΩ ΤΡΟΠΟ!

Η παράδοση του αρχείου θα γίνει με τη χρήση της πλατφόρμας ηλεκτρονικής μάθησης του τμήματος (http://www.icsd.aegean.gr/eclass).

Σε όλα τα αρχεία που θα δημιουργήσετε (πηγαίος κώδικας) θα πρέπει να αναφέρετε στην αρχή του κειμένου τα ονόματα των μελών της ομάδας εργασίας.

Υποδείξεις. Η εργασία είναι ***2 ατόμων.*** Τα προγράμματα πρέπει να υλοποιηθούν σε γλώσσα C ή C++.