ΕΚΤΕΛΕΣΗ

Στο directory HT_lib ή HP_lib αντίστοιχα:

- **make** : Παράγει το εκτελέσιμο 'main' το οποίο αποτελεί το demo κάθε δομής.
- **make run** : Παράγει το εκτελέσιμο 'main' και το εκτελεί.
- **make clean** : Διαγράφει όλα τα εκτελέσιμα και αντικειμενικά αρχεία που έχουν παραχθεί απο τη make.

ΓΕΝΙΚΑ

- Χρησιμοποιήθηκε η παράμετρος -std=c++11 για τη μεταγλώτισση.
- Χρησιμοποιήθηκε η παράμετρος -w για να αγνοηθούν τα warnings. Τα warnings αφορούν το εξής: Για να συμβαδίζουν οι ορισμοί την συναρτήσεων με αυτούς της εκφώνησης ορίσαμε τα arguments ως char* και όχι string, ενώ περνάμε μεταβλητές τύπου string.

HT_LIB

- Με την κλήση της `HT_CreateIndex` δημιουργούνται 2 αρχεία:
  - Το αρχείο που περιέχει τα blocks και τα περιεχόμενα τους και έχει όνομα filename το οποίο είναι το 1ο όρισμα της `HT_CreateIndex`.
  - Το αρχείο 'index' που περιέχει τους δείκτες σε κάθε bucket. Το αρχείο αυτο περιέχει τόσες γραμμές, όσες και το πλήθος των buckets. Κάθε γραμμή περιέχει έναν δείκτη σε ένα bucket. Το αποτέλεσμα της hash function είναι ένας αριθμός από το 1 έως το πλήθος των buckets του πίνακα κατακερματισμού. Αυτός ο αριθμός αποτελεί την γραμμή στην οποία βρίσκεται ο δείκτης του bucket στο οποίο θα τοποθετηθεί η εγγραφή.

    Σημείωση: Στην συγκεκριμένη περίπτωση με τη δημιουργία του hash table δεσμέυουμε κατευθείαν 1 block για κάθε bucket και εφόσον οι αριθμοί των buckets δημιουργούνται σειριακά τυχαίνει ο αριθμός γραμμής να είναι ίδιος με τον δείκτη του block. Ωστόσο δεν έχουμε εκμεταλλευτεί αυτή την ιδιότητα, γι'αυτο και υπάρχει το αρχείο index.
- Κάθε block αποτελείται από 5 records στα πρώτα 5*sizeof(Record) bytes και έναν ακέραιο στα τελευταία sizeof(int) bytes ο οποίος αποτελεί τον δείκτη στο επόμενο block ή -1 αν δεν υπάρχει επόμενο block. Οι συναρτήσεις `setNextBlock` και `getNextBlock` βοηθούν στο να θέσουμε και να πάρουμε αντίστοιχα τον ακέραιο αυτόν.
- Για την εισαγωγή εγγραφών ξεκινάμε από το 1ου block του bucket στο οποίο ανήκει η εγγραφή και ελέγχουμε για κάθε πιθανή θέση εγγραφής αν υπάρχει εγγραφή. Αν υπάρχει, ελέγχουμε αν έχει ίδιο id με αυτή που θέλουμε να εισάγουμε. Εφόσον δεν υπάρχει εγγραφή με το ίδιο id, την εισάγουμε στην πρώτη κενή θέση που βρίσκουμε, αναζητώντας ξανά από το 1ο block.
- Για την διαγραφή, εκτελούμε αναζήτηση της εγγραφής με τον ίδιο τρόπο. Αν βρεθεί η εγγραφή, γεμίζουμε αυτή τη θέση στο block με μηδενικά.

HP_LIB

Η υλοποίηση του σωρού έγινε με την χρήση ενός struct σε κάθε ένα block. Κάθε block λοιπόν περιέχει ένα struct και μόνο. Αυτό περιέχει έναν πίνακα που αποτελείται από 5 records (όσα μπορεί να χωρέσει ένα block συμφωνα με τους υπολογισμούς μας) , ένα int που περιγράφει το πόσα records υπάρχουν εκείνη της στιγμή στο block καθώς και ένα boolean value που είναι true αν έχει υπάρξει διαγραφή στο block . Τα δύο τελευταία μέλη του block είναι χρήσιμα στις υλοποιήσεις insert and delete . Οι υλοποιήσεις αυτές κάθε αυτές δεν είναι περίπλοκες. Η λογική της insert είναι ότι ψάχνει τα blocks ένα ένα αν έχουν διαθέσιμο χώρο τοσο επειδή δεν έχει έχει εισαχθεί ο μέγιστος αριθμός από records είτε έχει γιατί έχει υπάρξει διαγραφή και το record θα γίνει insert σε εκείνο το σημείο. Αν όλα τα blocks είναι γεμάτα , τότε δημιουργείται στο αρχείο ένα νέο block στο οποίο γίνεται insert το επικείμενο record. Η delete σε πρώτη φάση εκτελεί αναζήτηση στο file και αν υπάρχει το record που είναι προς διαγραφή , τότε κάνει το id του -1 και σε κάθε πεδίο του γράφει τη λέξη “DELETED” ή αν δεν υπάρχει εκτυπώνει μήνυμα λάθους. Η αλλαγή του id σε -1 αποσκοπεί στο να αναγνωρίζετε η εγγραφή αυτή ως διεγραμμένη και να είναι δυνατή η μελλοντική εκχώρηση κάποιας άλλης στη θέση της σε εκείνο το block. Τέλος, πρέπει να σημειώσουμε ότι η create δημιουργεί 2 blocks όπου το πρώτο περιέχει το απαραίτητο info για το heap και το επόμενο είναι το πρώτο που περιέχει records.