**Project 2 - B+ Tree**

**Συντελεστές** :

***ΕΠΙΘΕΤΟ : ΚΑΡΑΜΠΑΣ***

***ΟΝΟΜΑ : ΕΛΕΥΘΕΡΙΟΣ - ΑΡΓΥΡΙΟΣ***

***ΑΜ : 1115201400064***

***ΕΠΙΘΕΤΟ : ΚΟΥΚΟΥΛΗΣ***

***ΟΝΟΜΑ: ΑΘΑΝΑΣΙΟΣ***

***ΑΜ: 1115201400328***

Για την υλοποίηση της εργασίας ακολουθήθηκε η εξής δομή για τα αρχεία :

1. **include :** Περιέχει όλα τα αρχεία κεφαλίδας που χρειάζονται .

|  |  |
| --- | --- |
| Φάκελος include | |
| **Όνομα Αρχείου** | **Περιγραφή** |
| AM.h | περιέχει τις δηλώσεις συναρτήσεων του επιπέδου ευρετηρίου ΑΜ |
| B\_Tree.h | περιέχει τις δηλώσεις συναρτήσεων οι οποίες χρησιμοποιήθηκαν ως βοηθητικές για την υλοποίηση των συναρτήσεων του επιπέδου ευρετηρίου |
| Global\_Struct.h | περιλαμβάνει τις δηλώσεις των global δομών που δημιουργήσαμε όπως η Open\_Files και η Scan\_Files |
| List.h | περιέχει τη δήλωση της δομής λίστας που υλοποιήσαμε καθώς και τα πρώτυπα των συναρτήσεων που την αποτελούν |
| defn.h | ---------- |
| bf.h | ---------- |

#### Global Structures

##### Open\_Files : Δομή που αντικατοπτρίζει τα ανοιχτά αρχεία και περιλαμβάνει σημαντικές πληροφορίες για το αρχείο όπως το αναγνωριστικό του (fileDesciptor) , τους τύπους των πεδίων του όπως επίσης και τη ρίζα του δέντρου .

##### Scan\_Files : Δομή που αντικατοπτρίζει τα αρχεία με ανοιχτές σαρώσεις . Περιλαμβάνει πληροφορίες σχετικά με το που θα εκτελεστεί μια πράξη σύγκρισης (συγκεκριμένο μπλοκ και θέση μέσα σε αυτό) , ποια θα είναι η πράξη , όπως επίσης και ποια θα είναι η τιμή που θα συγκριθεί με την τιμή που έχουμε βρει απο το μπλοκ .

1. **src :** Περιέχει τον βασικό κορμό της άσκησης .

|  |  |
| --- | --- |
| Φάκελος src | |
| **Όνομα Αρχείου** | **Περιγραφή** |
| *AM.c* | *Βασικές συναρτήσεις* |
| *B\_Tree.c* | *Βοηθητικές συναρτήσεις* |
| *List.c* | Στο αρχείο αυτό έχουν υλοποιηθεί οι βασικές συναρτήσεις της λίστας (ως stack) , δηλαδή συναρτήσεις δημιουργίας , εισαγωγής , εξαγωγής και καταστροφής. |
| *Main.c* | *Extra main (for testing purposes)* |

**Στον παρακάτω πίνακα παρατίθενται οι βασικές συναρτήσεις της άσκησης :**

|  |  |
| --- | --- |
| Αρχείο AM.c - βασικές συναρτήσεις | |
| **Όνομα Συνάρτησης** | **Περιγραφή** |
| AM\_Init | Αρχικοποιεί τον επίπεδο BF καθώς και δημιουργεί τις global δομές μας |
| ΑΜ\_CreateIndex | Αρχικά γίνεται έλεγχος για το αν τα ορίσματα που λαμβάνουμε είναι εντός των προδιαγραφών . Έπειτα αρχικοποιούμε το πρώτο μπλοκ εισάγοντας στα πρώτα byte το αναγνωριστικό “B+” ώστε να γνωρίζουμε οτι πρόκειται για αρχείο ευρετηρίου Β+ δέντρου όπως επίσης και πληροφορίες σχετικά με τα πεδία των εγγραφών και τη θέση της ρίζας (αριθμός μπλοκ) . |
| AM\_DestroyIndex | Αναζήτηση στον πίνακα των ανοιχτών αρχείων με βάση το filename . Αν δεν υπάρχει το αρχείο filename ως ανοιχτό , διαγράφουμε το αρχείο από τον δίσκο . |
| AM\_OpenIndex | Ανοίγουμε ένα αρχείο και ελέγχουμε αν είναι αρχείο Β+ . Αν είναι , ψάχνουμε να βρούμε κενή θέση στον πίνακα για να εισάγουμε το νέο αρχείο που ανοίξαμε . Στην περίπτωση που βρούμε κενή θέση , αντιγράφουμε από το πρώτο μπλοκ του αρχείου (metadata) τις πληροφορίες σχετικά με τα πεδία των εγγραφών και τη θέση της ρίζας (αριθμός μπλοκ) στο struct open\_files καθώς και το όνομα του αρχείου . |
| AM\_CloseIndex | Αρχικά ελέγχουμε αν υπάρχει ανοιχτή σάρωση στο αρχείο που θέλουμε να κλείσουμε . Αν βρούμε κάποια , τότε επιστρέφουμε μήνυμα λάθους , αλλιώς κλείνουμε το αρχείο . |
| AM\_Insert | Αρχικά ελέγχουμε αν έχει δημιουργηθεί η ρίζα του δέντρου , αν όχι τότε την δημιουργούμε . Στη συνέχεια καλούμε την συνάρτηση traverse εως ότου φτάσουμε σε data block , αποθηκεύοντας παράλληλα τα μπλοκ που διασχίζουμε σε μια λίστα . Η λίστα αυτή θα μας χρησιμεύσει στην περίπτωση που θα χρειαστεί να κάνουμε split ένα μπλοκ . ***α)*** Αφού γνωρίζουμε το μπλοκ που θα πρέπει να εισάγουμε την νέα τιμή που μας έχει δοθεί τότε με την βοήθεια της συνάρτησης sort έλεγχουμε αν χωράει η τιμή μας στον υπάρχων μπλοκ και αν ναι την εισάγει και ταξινομεί το μπλοκ κατάλληλα . Αν η νέα τιμή δεν χωράει τότε προχωράμε σε σπάσιμο του μπλοκ και στην εισαγωγή της . Το βήμα ***α)*** θα καλείται επαναληπτικά εως ότου δεν χρειάζεται να ξανασπάσουμε κάποιο μπλοκ από τις πιο πάνω βαθμίδες . |
| AM\_OpenIndexScan | Αρχικά ελέγχουμε αν υπάρχει ανοιχτή σάρωση στο αρχείο που θέλουμε να κλείσουμε . Αν βρούμε κάποια , τότε επιστρέφουμε μήνυμα λάθους , αλλιώς κλείνουμε το αρχείο . |
| AM\_FindNextEntry | Διαβάζουμε από την δομή Scan\_Files τις πληροφορίες που χρειάζονται για την σάρωση . Αποθηκεύουμε την τιμή του αποτελέσματος που θα επιστρέψουμε και έπειτα αναζητούμε την επόμενη θέση που πληρεί τα κριτήρια αν αυτή υπάρχει . |
| AM\_CloseIndexScan | Κοιτάμε αν υπάρχει η θέση scanDesc μέσα στον πίνακα μας . Αν υπάρχει διαγράφουμε την δομή . |
| AM\_Close | Καταστρέφει τις global δομές . |

**Στον παρακάτω πίνακα παρατίθενται οι βοηθητικές συναρτήσεις της άσκησης:**

|  |  |
| --- | --- |
| Αρχείο B\_Tree.c - Βοηθητικές συναρτήσεις | |
| **Όνομα Συνάρτησης** | **Περιγραφή** |
| Initialize\_Root | Δημιουργία ρίζας, ενημέρωση ρίζας πρώτου μπλοκ (metadata) και ενημέρωση του struct ανοιχτών αρχείων. |
| compare | Παίρνει δύο τιμές (void \*) και επιστρέφει 0 αν η αλλιώς 1 . |
| op\_function | Παίρνει δύο τιμές καθώς και ένα τελεστή σύγκρισης op και επιστρέφει 0 αν ισχύει αλλιώς 1 , όπου |
| sort | Προσπαθεί να εισάγει ένα ζευγάρι κλειδιού - τιμής (η τιμή μπορεί να είναι είτε δείκτης σε μπλοκ είτε τιμή πεδίου) σε ένα μπλοκ και παράλληλα να το ταξινομήσει . Αν χωράει ολοκληρώνει το έργο της αλλιώς επιστρέφει -1 . |
| traverse | Ανοίγουμε το μπλοκ, κοιτάμε αν είναι μπλοκ δεδομένων ή μπλοκ ευρετηρίου .   1. Μπλοκ δεδομένων : Επιστρέφει -1 ώστε να σταματήσει η διάσχιση στο δέντρο 2. Μπλοκ ευρετηρίου: Αναζήτηση για το κατάλληλο μονοπάτι που θα πρέπει να ακολουθήσουμε . Στην περίπτωση που το μονοπάτι ισούται με -1 (NULL) τότε δημιουργούμε καινούριο μπλοκ , το αρχικοποιούμε και κάνουμε τις κατάλληλες ενώσεις με τα γειτονικά μπλοκ (δεξιά - αριστερά) αν αυτά υπάρχουν και επιστρέφεται ο αριθμός του καινούριου μπλοκ . Αλλιώς επιστρέφεται το μονοπάτι . |
| split | Ανοίγει ένα καινούριο μπλοκ και το αρχικοποιεί. Στη συνέχεια κοιτάει αν είναι μπλοκ δεδομένων ή ευρετηρίου . Αν είναι μπλοκ δεδομένων κοιτάει την περίπτωση να υπάρχουν περισσότερες από μια ίδιες τιμές και προσπαθεί να κάνει το καλύτερο δυνατό σπάσιμο . Αν είναι μπλοκ ευρετηρίου σπάει στη μέση το παλιό μπλοκ και μοιράζει τις τιμές και βάζει και την καινούρια τιμή στη σωστή θέση . |
| Find\_ScanIndex\_position | Βρίσκει την πρώτη αποδεκτή θέση του μπλοκ με βάση τον δοσμένο τελεστή σύγκρισης . |
| Find\_Scan | Αν βρισκόμαστε σε μπλοκ δεδομένων παίρνουμε με την βοήθεια της Find\_ScanIndex\_position τον αριθμό του μπλοκ και τη θέση του record\_number στο μπλοκ αυτό . Αλλιώς αν είμαστε σε μπλοκ ευρετηρίου , διακρίνουμε τις περιπτώσεις ανάλογα με τον τελεστή σύγκρισης που έχουμε ώστε να διασχίσουμε το δέντρο . |