

3^η Άσκηση ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Διαχείριση Στοιχείων Αεροδρομίων με ΑΤΔ Ευρετήριο (χρήση ΔΔΑ)

Να κατασκευάσετε τον ΑΤΔ Ευρετήριο και να τον χρησιμοποιήσετε για την διαχείριση των εξής στοιχείων αεροδρομίων: κωδικός ICAO (4 γράμματα), κωδικός IATA (3 γράμματα), τόπος, όνομα αεροδρομίου και Χώρα, π.χ

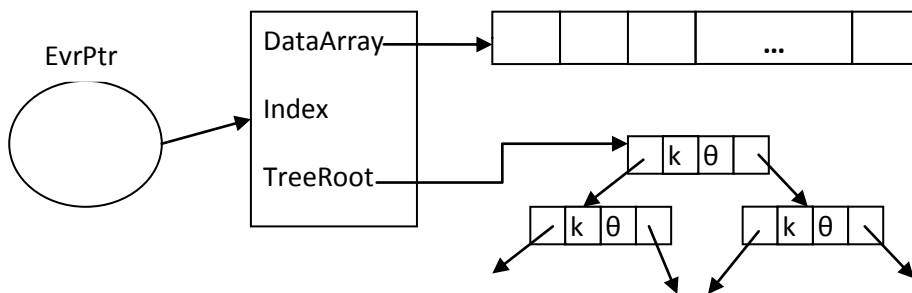
SVSP	SNF	San Felipe	Sub Teniente Nestor Arias	Venezuela
------	-----	------------	---------------------------	-----------

[ICAO-International Civil Aviation Organization, is a branch of the United Nations, represents aviation authorities of UN member nations. IATA-International Air Transport Association, represents most major scheduled airlines, deals with commercial aspect of airline operations, i.e. ticketing, interline baggage transfer, liability limits, etc.]

A. Ο ΑΤΔ Ευρετήριο (Evr)

Αποτελείται από 1) έναν πίνακα, όπου αποθηκεύονται τα στοιχεία των αεροδρομίων 2) έναν ακέραιο που δείχνει το πρώτο διαθέσιμο κελί στον πίνακα, και 3) Δυαδικό Δένδρο Αναζήτησης, όπου αποθηκεύονται α) ο κωδικός ICAO, που είναι το κλειδί (key) του ΔΔΑ και β) η θέση (int) στον πίνακα με τα πλήρη δεδομένα του αεροδρομίου.

```
typedef struct EvrNode * EvrPtr;
struct EvrNode{
    typos_deikti TreeRoot;          /* Root of Tree */
    TStoixeiouOther *dataArray;      /* array of size MaxSize */
    int Index;                       /* index of first available element in array */
};
```



Ακολουθούν οι πράξεις του Evr

`EvrPtr EvrConstruct(int MaxSize);` Δημιουργεί έναν κόμβο `EvrNode` και αρχικοποιεί τα 3 μέλη του. Ειδικά για το `dataArray` δημιουργεί δυναμικά έναν πίνακα `MaxSize` στοιχείων. Πίνακας με 5000 στοιχεία είναι αρκετός.

`int EvrDestruct(EvrPtr *E);` Καταστρέφει την Δομή `E` (όλα τα στοιχεία του δένδρου, τον πίνακα και τον κόμβο `EvrNode`). Επιστρέφει ένδειξη αν εντοπίσει λάθος.

`int EvrInsert(EvrPtr E, TStoixeiouEvr Data);` Εισάγει στο πρώτο ελεύθερο στοιχείο του πίνακα τα δεδομένα `Data`, και εισάγει στο ΔΔΑ το κλειδί `key` μαζί με την θέση, όπου εισήχθησαν τα `Data`. Επιστρέφει ένδειξη αν εντοπίσει λάθος.

`int EvrSearch(EvrPtr E, keyType key, int *found);` Αναζητά στην `E` (ΔΔΑ) το κλειδί `key` και ανάλογα θέτει τιμή στο `found`. Επιστρέφει ένδειξη αν εντοπίσει λάθος.

`int EvrPrintAll(EvrPtr E, FILE *out, int Counter);` Εκτυπώνει στο `out` όλα τα στοιχεία με αλφαβητική σειρά των κλειδιών (ενδοδιάταξη). Υπολογίζει τα στοιχεία που επισκέφτηκε. Επιστρέφει ένδειξη αν εντοπίσει λάθος.

B. Διαχείριση Στοιχείων Αεροδρομίων

Να χρησιμοποιήσετε τον ΑΤΔ Ευρετήριο σε πρόγραμμα πελάτη `AirportManagement.c` για την διαχείριση στοιχείων αεροδρομίων. Συγκεκριμένα στο πρόγραμμα πελάτη:

1) Δημιουργήστε ένα Ευρετήριο.

2) Εισάγετε στοιχεία στο Ευρετήριο (ΔΔΑ+πίνακα) από το αρχείο `ICAO_Random_Input.txt` που περιέχει τα στοιχεία των αεροδρομίων με τους κωδικούς ICAO σε τυχαία διάταξη. Τα στοιχεία α) να τα εισάγετε στον πίνακα και β) να εισάγετε στο ΔΔΑ τον κωδικό ICAO (κλειδί) μαζί με την θέση του πίνακα, όπου αποθηκεύτηκαν τα στοιχεία. Επίσης να μετράτε και να εμφανίζετε τον χρόνο εισαγωγής μετά από $(2^N - 1)$ στοιχεία, όπου $N=9,10,11,12$, καθώς τον ολικό αριθμό στοιχείων και και τον ολικό χρόνο που απαιτήθηκε. Τα στοιχεία κάθε αεροδρομίου είναι σε μια γραμμή του αρχείου και διαχωρίζονται με το σύμβολο `%`. Οι πρώτες γραμμές του αρχείου είναι:

```
SVSP%SNF%San Felipe%Sub Teniente Nestor Arias%Venezuela
ENDU%BDU%Bardufoss%Bardufoss Airport%Norway
```

FMND%ZWA%Andapa%Andapa%Madagascar
VNGK%GKH%Gorkha%Gorkha%Nepal
CYGX%YGX%Gillam, Manitoba%Gillam%Canada

3) Αναζήτηση δεδομένων από το αρχείο ICAO_SEARCH.txt που περιέχει κωδικούς ICAO, έναν σε κάθε γραμμή. Οι πρώτες γραμμές του αρχείου:

AGAF
AGAR
AGAT
AGBA
AGBT

α) Να διαβάσετε τους ICAO από το αρχείο και να διαπιστώνετε αν υπάρχουν στο ΔΔΑ.

β) Εκτυπώστε τον μέσο χρόνο αναζήτησης και πόσα ICAO βρέθηκαν και πόσα δεν βρέθηκαν.

γ) Εκτυπώστε στο αρχείο ICAO_FOUND.txt τους κωδικούς ICAO που βρέθηκαν στο ΔΔΑ.

4) Να επαναλάβετε το 3) με είσοδο το αρχείο ICAO_FOUND

5) Να εκτυπώσετε στο αρχείο AIRPORTS.txt αλφαβητικά με βάση τον κωδικό ICAO τα δεδομένα (κλειδί+υπόλοιπα) καθώς και τον αριθμό των στοιχείων που εκτύπωσε και τον συνολικό χρόνο που απαιτήθηκε για την εκτύπωση.

6) Να καταστρέψετε το Ευρετήριο.

7) Να επαναλάβετε τα ζητούμενα 1-6, διαβάζοντας όμως κωδικούς από το ICAO_Sorted_Input.txt, που περιέχει κωδικούς ICAO ταξινομημένους σε αύξουσα σειρά.

Να συμπληρώσετε τον ακόλουθο πίνακα σύμφωνα με τις μετρήσεις σας (στα κενά βάζετε χρόνους, όπου «μέσος όρος» τον αντίστοιχο μέσο όρο, και στα σκιασμένα κελιά τον αντίστοιχο αριθμό στοιχείων).

Εργασία/Χρόνος	μέγεθος	Απλό ΔΔΑ random input	Απλό ΔΔΑ sorted input	AVL ΔΔΑ random input	AVL ΔΔΑ sorted input
Εισαγωγές (Ερ. 2)	511				
	1023				
	2047				
	4095				
	Όλα				
	Όλα	Μέσος όρος	Μέσος όρος	Μέσος όρος	Μέσος όρος
Αναζητήσεις (Ερ.3)	Όλα	Μέσος όρος	Μέσος όρος	Μέσος όρος	Μέσος όρος
(Ερ. 4)	Βρέθηκαν	Μέσος όρος	Μέσος όρος	Μέσος όρος	Μέσος όρος
Εκτύπωση (Ερ. 5)	Όλα				

Σύντομο; σχολιασμό των χρόνων σε σχέση με τους θεωρητικούς χρόνους (π.χ. $O(n)$, $O(\log n)$, $O(n^2)$)

Να μεταγλωττίσετε και εκτελέσετε το κυρίως πρόγραμμα με τις δυο υλοποιήσεις του Ενρ (BST και AVL).

Απαιτείται επίσης να αναπτύξετε τύπους στοιχείων TSEnr, TSDDA με οδηγό τους τύπους που σας έχουν δοθεί.

Οδηγίες Σχεδίασης και Ανάπτυξης Προγράμματος

Σας δίδετε σκελετός προγράμματος, που απαιτείται να αναπτύξετε. Το πρόγραμμά σας να είναι οργανωμένο σε ενότητες (modules) και σε πρόγραμμα-πελάτη. Να αναπτύξετε το AirportManagement.c, την υλοποίηση του Ενρ.h στο Ενρ.c (ολική απόκρυψη με δύο εναλλακτικές υλοποιήσεις για απλό BST και AVL), TSEnr.h, TSEnr.c, TSDDA.h και TSDDA.c.

Οι πράξεις του Ενρ.c να υλοποιηθούν χρησιμοποιώντας πράξεις των ΑΤΔ ΔΔΑ και ΑΤΔ AVL (μία διεπαφή Ενρ.h και δυο υλοποιήσεις στο ίδιο αρχείο Ενρ.c με χρήση #if ...#else...#endif – δείτε τον κώδικα). Οι υλοποιήσεις των ΑΤΔ ΔΔΑ και AVL σας δίνονται έτοιμες στα αντίστοιχα ζεύγη αρχείων (BST.c, BST.h) και (AVL.c, AVL.h) σε υποφακέλους BST και AVL, αντίστοιχα. **Δεν πρέπει να επεμβείτε στον κώδικά τους.**

Ως οδηγό της υλοποίησης μπορείτε να δείτε την τεχνική της υλοποίησης ουράς με χρήση του ΑΤΔ Λίστα στις διαφάνειες με τις εφαρμογές τις λίστας (7.8.3 των πρότυπων υλοποιήσεων). Οι δυο υλοποιήσεις BST και AVL έχουν παρόμοια διεπαφή. Οι διαφορές είναι α) ότι οι πράξεις του απλού BST έχουν το πρόθεμα Tree ενώ του AVL το

AVLTree και β) οι πράξεις εισαγωγής και διαγραφής σε AVL χρειάζονται μια *int παράμετρο επιπλέον (που δεν χρειάζεται αρχική τιμή κατά την κλήση των συναρτήσεων). Δεν πρέπει να αλλάξετε την υλοποίηση των BST, AVL ή τα ονόματα των αρχείων δεδομένων. Το πρόγραμμά σας θα ελεγχθεί με αυτά τα αρχεία.

Οι πράξεις του Enr.h να περιέχουν συμβόλαια (contracts) υλοποιημένα με assert και να εντοπίζουν λάθη. Το κυρίως πρόγραμμά σας να είναι οργανωμένο σε συναρτήσεις που να αντιστοιχούν στα ερωτήματα (ιδίως τα 2, 3, 4, 5) της άσκησης, ώστε το σώμα της main να είναι σύντομο. Να ελέγχετε την εγκυρότητα των δεδομένων, ενδείξεις λάθους από συναρτήσεις βιβλιοθηκών, του Enr.h, κλπ. Αναπτύξτε και ελέγξτε τις πράξεις του Enr σε ξεχωριστό πρόγραμμα test.c πριν απαντήσετε στα ερωτήματα 2 έως 5.

Για την χρονομέτρηση σας δίνονται στον κατάλογο TimingDemos τρία προγράμματα επίδειξης: το SecResolutionTiming.c χρησιμοποιεί την βιβλιοθήκη της C με ακρίβεια δευτερολέπτου και τα άλλα δύο WinMilliSecResTiming.c και LinuxMilliSecResTiming.c βιβλιοθήκες των Windows και Linux αντίστοιχα για ακρίβεια msec, που μάλλον θα σας χρειαστεί για μεγαλύτερη ακρίβεια μετρήσεων.