Άσκηση 1η : Πρόγραμμα υλοποίησης ΑΑ Δένδρου

Στο πρόγραμμα “AA.cpp” έχουν υλοποιηθεί, εκτός από τις 3 βασικές λειτουργίες που ζητήθηκαν (insert, delete, search) και άλλες ακόμα, βοηθητικές λειτουργίες και συναρτήσεις, οι οποίες συνεισφέρουν στην ομαλή λειτουργία και ευκολότερη κατανόηση του κώδικα.

Ξεκινάμε, λοιπόν, ορίζοντας το macro \_\_min το οποίο επιστρέφει τη μικρότερη από δύο τιμές και το οποίο θα χρησιμοποιηθεί αργότερα. Ορίζουμε έπειτα τον τύπο των δεδομένων που περιέχει το ΑΑ δένδρο ως ακεραίους και έπειτα ορίζουμε το struct που αντιπροσωπεύει έναν κόμβο του δένδρου με τα χαρακτηριστικά του. Έπειτα κάνουμε την δήλωση των μεθόδων που θα χρησιμοποιήσουμε, πριν την υλοποίησή τους, για τη σωστή λειτουργία του προγράμματος.

Ξεκινάμε, τώρα, με την υλοποίηση των διαφόρων μεθόδων, με πρώτη την mallocNode, η οποία διανέμει δυναμικά μνήμη για τον κάθε νέο κόμβο του δένδρου. Έπειτα ακολουθούν οι συναρτήσεις Skew και Split, συναρτήσεις απαραίτητες για την υλοποίηση ενός ΑΑ δένδρου, με την Skew να διορθώνει τυχών προβλήματα των οριζόντιων αριστερών δεικτών και την Split να διορθώνει τυχών προβλήματα των διαδοχικών δεξιών δεικτών.

Στη συνέχεια, ακολουθούν οι συναρτήσεις Predecessor και Successor, οι οποίες είναι υπεύθυνες για τις κινήσεις που απαιτούνται από των «πατέρα» και από το «παιδί» του τρέχων κόμβου, αντίστοιχα.

Ακολουθεί η συνάρτηση Decrease\_Level η οποία, όπως δηλώνει και το όνομα, μειώνει το επίπεδο ενός κόμβου όταν τηρούνται οι απαραίτητες συνθήκες που έχουμε θέσει (όπως το να πάψει να έχει κόμβος 2 παιδιά ενώ βρίσκεται σε επίπεδο μεγαλύτερο του επιπέδου 1), ώστε να διατηρηθεί η ορθή δομή του δένδρου και να μην παραβιαστούν οι ιδιότητές του.

Έπειτα, υλοποιούμε τη μέθοδο \_\_InsertData που εισάγει έναν νέο κόμβο, με το κλειδί του και την τιμή του στο ΑΑ δένδρο και τη συνάρτηση \_\_DeleteData που διαγράφει έναν κόμβο βάση του κλειδιού του, με τη χρήση φυσικά των μεθόδων Skew, Spit, Predecessor και Successor, όπου αυτές χρειάζονται. Για την κλήση των δύο αυτών μεθόδων έχουμε φτιάξει άλλες δύο σύντομες μεθόδους, μία για την καθεμία, τις aat\_insertData και att\_deleteData, αντίστοιχα, οι οποίες καλούν τις δύο προηγούμενες μεθόδους, κάνοντας την main μέθοδο πιο ευανάγνωστη και των κώδικα πιο «καθαρό».

Ακολουθεί η 3η βασική συνάρτηση που απαιτείται στις προδιαγραφές, η aat\_searchData η οποία, φυσικά, αναζητάει ένα στοιχείο μέσα στο δένδρο βάση του κλειδιού του, ακολουθώντας τις κινήσεις που μας είναι γνωστές σε ένα οποιοδήποτε Δυαδικό Δένδρο Αναζήτησης. Η πρώτη από τις δύο τελευταίες συναρτήσεις είναι η att\_deleteTree, η οποία διαγράφει ολόκληρο το ΑΑ δένδρο, διαγράφοντας αναδρομικά έναν-έναν τους κόμβους του, ελευθερώνοντάς τους και θέτοντάς τους ως NULL. Η τελευταία συνάρτηση είναι η att\_printTree η οποία εκτυπώνει ολόκληρο το ΑΑ δένδρο με όλους τους κόμβους που βρίσκονται στη συγκεκριμένη χρονική στιγμή μέσα σε αυτό, έχοντας μάλιστα μια στοιχειώδη μορφή που κατά την εκτύπωση θυμίζει δυαδικό δένδρο.

Στο τέλος, υπάρχει ο εξαιρετικά λυτός κώδικας της main, όπου δημιουργούμε το δένδρο και εκτελούμε στη σειρά τις διαδικασίες insert, delete και search, όσες φορές και για όποια στοιχεία θέλει ο χρήστης. Τέλος, εκτυπώνουμε το ΑΑ δένδρο ως έχει διαμορφωθεί μετά τις ενέργειες του χρήστη και τελικά το διαγράφουμε, ελευθερώνοντας τη μνήμη μου καταλαμβάνει.

Παναγιώτης Λειβαδάρος

Π21085