ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ
ΕΡΓΑΣΙΑΣ ΔΟΜΩΝ ΔΕΔΟΜΕΝΩΝ
2020-2021

(Υλοποιήθηκε από τους:
Όμηρος Μαλάκης, ΑΕΜ:3742
Σωτήριος Λουκάς Καμπύλης, ΑΕΜ:3805)

- Γενικά Θέματα-Main:

  Οι βιβλιοθήκες που αξιοποιεί η main, εκτός από την iostream, είναι οι: string, fstream, chrono και ctime. Για τις ανάγκες της εργασίας, οι λέξεις που εισάγει το πρόγραμμα μας στις πέντε δομές δεδομένων αποτελούνται μόνο από πεζούς λατινικούς χαρακτήρες. Η διαδικασία εξαγωγής λέξεων από το αρχείο είναι η εξής: Για κάθε γραμμή που περιέχει το αρχείο, η γραμμή αυτή αποθηκεύεται σε μία συμβολοσειρά line. Στην συνέχεια, χρησιμοποιούμε μια μεταβλητή pos, στην οποία αποθηκεύεται η θέση της πρώτης εμφάνισης του χαρακτήρα του κενού στην line με την χρήση της μεθόδου find. Χρησιμοποιώντας την pos και την μέθοδο substr αποθηκεύουμε σε μία νέα συμβολοσειρά, την word, την υποσυμβολοσειρά της line που ξεκινά από τον πρώτο της χαρακτήρα μέχρι τον χαρακτήρα που προηγείται του πρώτου κενού. Έπειτα, διαγράφουμε από την line την παραπάνω υποσυμβολοσειρά της συν το κενό που την ακολουθεί, εκτός αν το pos είναι ίσο με το npos της line, που σημαίνει ότι όλη η συμβολοσειρά έχει “καταναλωθεί”, στην οποία περίπτωση κάνουμε την line κενή συμβολοσειρά για να βγούμε από τον βρόγχο while. Πριν εισαχθεί στις δομές, η συμβολοσειρά word περνιέται στην συνάρτηση RemoveSpecialCharacters, η οποία διαγράφει τους χαρακτήρες που δεν είναι γράμματα και μετατρέπει τους χαρακτήρες που είναι γράμματα σε πεζά, και ελέγχεται αν είναι διάφορη της κενής συμβολοσειράς. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να τελειώσουν οι γραμμές του αρχείου. Η τυχαία επιλογή των λέξεων τις οποίες αναζητούμε στις δομές γίνεται με την βοήθεια των σταθερών SIZE

και PROBABILITY και της rand(). Η SIZE είναι ο μέγιστος αριθμός τυχαίων λέξεων και η πιθανότητα να εισαχθεί μία λέξη στον πίνακα των τυχαίων λέξεων είναι 1/PROBABILITY. Αφότου εισαχθεί μία λέξη στις δομές, ελέγχεται αν ο πίνακας των τυχαίων λέξεων έχει διαθέσιμο χώρο. Αν έχει, τότε η λέξη εισάγεται στον πίνακα αν και μόνο αν το υπόλοιπο της διαίρεσης του τυχαίου αριθμού που παράγει η rand() με την σταθερά PROBABILITY είναι ίσο με 1. Οι τιμές που δίνουμε στην SIZE και PROBABILITY είναι 10.000 και 25 αντίστοιχα, το οποίο έχει ως αποτέλεσμα λίγο λιγότερες από 10.000 τυχαίες λέξεις (για το κείμενο του Ντα Βίντσι), αλλά μπορείτε να αλλάξετε τις σταθερές αυτές για να δοκιμάσετε έναν άλλο αριθμό τυχαίων λέξεων. Ένα μειονέκτημα αυτής της υλοποίησης που πρέπει να αναφέρουμε είναι ότι οι λέξεις όπως οι of, the(η οποία είναι και ρίζα του BST στο κείμενο του Ντα Βίντσι) κτλ, επειδή εμφανίζονται πολύ περισσότερες φορές από τις άλλες λέξεις, έχουν μεγαλύτερη πιθανότητα να εισαχθούν στον πίνακα των τυχαίων λέξεων πολλές φορές. Έτσι το BST έχει ένα πλεονέκτημα στην αναζήτηση των λέξεων αυτών έναντι του AVL, καθώς οι λέξεις αυτές βρίσκονται πολύ κοντά στην ρίζα του, αφού εμφανίζονται από την αρχή του κειμένου. Για το benchmarking των δομών αποφασίσαμε να υλοποιήσουμε δύο μεθόδους αναζήτησης σε κάθε δομή, μία που απλώς επιστρέφει true ή false, ανάλογα με το αν βρέθηκε η λέξη ή όχι, και μία που εκτός από το να επιστρέφει true ή false, αν βρεθεί η λέξη την εκτυπώνει και σε ένα αρχείο κειμένου. Έτσι, χρησιμοποιώντας την chrono και τις σταθερές TIME_UNIT και TIME_UNITS(που μπορείτε να τις αλλάξετε), μετράμε τον χρόνο που πήρε η κάθε δομή για να αναζητήσει τις τυχαίες λέξεις και εκτυπώνουμε τον αντίστοιχο χρόνο της κάθε δομής, καθώς και τον αριθμό των λέξεων που αναζητήθηκαν στο αρχείο “Results.txt”. Μετά αναζητούμε και πάλι τις τυχαίες λέξεις σε κάθε δομή, αλλά αυτή την φορά με την αναζήτηση που εκτυπώνει την κάθε λέξη σε ένα αρχείο κειμένου. Αυτό εξασφαλίζει πως χρονομετρούμε μόνο τον χρόνο που παίρνει η κάθε δομή να αναζητήσει τις λέξεις και όχι και τον χρόνο που παίρνει το σύστημα να εκτυπώσει τα αποτελέσματα σε αρχεία. Τέλος, θέλαμε να αναφέρουμε ότι χρησιμοποιήσαμε το memory testing tool Valgrind στο πρόγραμμα μας και προέκυψε πως η

διαχείριση της μνήμης γίνεται σωστά και δεν δημιουργούνται memory leaks.

- **ΚΛΑΣΗ: UnorderedArray**:
  Η UnorderedArray χρησιμοποιεί τις βιβλιοθήκες iostream, fstream και string. Μέσα στο αρχείο header της κλάσης υπάρχει η κλάση ArrayObj η οποία χρησιμοποιείται για να υλοποιήσει τα στοιχεία του πίνακα. Η ArrayObj έχει ως ιδιότητες μία συμβολοσειρά word που είναι λέξη του στοιχείου, έναν ακέραιο occurrences που είναι ο αριθμός των εμφανίσεων της λέξης του στοιχείου. Ο κατασκευαστής της ArrayObj παίρνει ως όρισμα μία συμβολοσειρά την οποία την αναθέτει στην word και αρχικοποιεί την τιμή των occurrences με 1. Στο private κομμάτι της UnorderedArray υπάρχει ο πίνακας UArray από στοιχεία ArrayObj, ο οποίος είναι ένας δυναμικός πίνακας από δείκτες σε αντικείμενα και αποτελεί την βάση της δομής. Επιπλέον υπάρχει η ιδιότητα size που είναι ένας ακέραιος αριθμός που ορίζει πόσο μεγάλος μπορεί να είναι ο πίνακας και η ιδιότητα counter που είναι ένας ακέραιος αριθμός που μετράει πόσα στοιχεία είναι μέσα στον πίνακα UArray. Οι μέθοδοι της κλάσης είναι:

-Κατασκευαστής: Ο κατασκευαστής θέτει στην μεταβλητή size το μέγεθος του πίνακα της κλάσης, αρχικοποιεί την μεταβλητή counter και δημιουργεί έναν νέο πίνακα κατασκευασμένο από δείκτες, οι οποίοι δείχνουν στο αντικείμενο της κλάσης (ArrayObj). Επίσης οι συγκεκριμένοι δείκτες αρχικοποιούνται με NULL.

-Καταστροφέας: Ο καταστροφέας διαγράφει το περιεχόμενο κάθε θέσης που δεν είναι κενή(NULL). Έπειτα διαγράφει ολόκληρο τον πλέον κενό πίνακα.

-void Insert(string): Η μέθοδος παίρνει ως όρισμα την λέξη string που θέλει το πρόγραμμα να καταχωρηθεί στον πίνακα. Η μέθοδος κάνει προσπέλαση τον στοιχείων του πίνακα και ελέγχει ένα προς ένα άμα η string που δόθηκε ταιριάζει με την word κάποιου στοιχείου του πίνακα. Άμα βρεθεί ότι κάποιο στοιχείο υπάρχει ήδη μέσα στον πίνακα με την λέξη που δόθηκε, ανεβάζουμε τα occurances του στοιχείου κατά 1 και τελειώνει η μέθοδος.

Αν κανένα στοιχείο του πίνακα δεν ταιριάζει με την λέξη που δόθηκε η μέθοδος πρώτα εξετάζει αν ο counter είναι μικρότερος του size. Σε περίπτωση false σημαίνει ότι ο πίνακας δεν έχει χώρο για καινούργια λέξη και εμφανίζεται ένα αντίστοιχο μήνυμα. Αλλιώς πηγαίνει στην θέση counter όπου είναι η πρώτη διαθέσιμη θέση και σε αυτή αρχικοποιεί ένα καινούργιο στοιχείο ArrayObj που περιέχει την λέξη string. Το counter ανεβαίνει κατά ένα.

-bool Search(string): Η μέθοδος κάνει προσπέλαση των στοιχείων ένα προς ένα με σειριακή αναζήτηση. Άμα βρεθεί κάποιο στοιχείο που να ταιριάζει με το string που δίνεται επιστρέφει true. Άμα φτάσει στο τέλος του πίνακα και δεν έχει βρεθεί το στοιχείο επιστρέφει false.

-bool Delete(string): Η μέθοδος αυτή είναι ίδια με την παραπάνω Search, όμως άμα βρεθεί το αναζητούμενο στοιχείο το διαγράφει και μειώνει το counter κατά ένα. Αν ολοκληρωθεί η διαδικασία επιτυχώς επιστρέφει true, διαφορετικά επιστρέφει false.

-bool PrintSearch(string): Η μέθοδος αυτή είναι ίδια με την παραπάνω Search, με την εξαίρεση ότι όταν βρίσκει την λέξη, εκτός από το να επιστρέφει true, την εκτυπώνει μαζί με τον αριθμό εμφανίσεων της στο αρχείο το οποίο δέχεται ως όρισμα.

- **ΚΛΑΣΗ: OrderedArray:**

  Αυτή η κλάση παρουσιάζει αρκετές ομοιότητες με την UnorderedArray, αποφασίσαμε όμως να μην ενώσουμε με κληρονομικότητα την μια με την άλλη καθώς υπάρχουν αρκετές διαφορές στις μεθόδους και για να μπορούν να χρησιμοποιηθούν ανεξάρτητα. Η OrderedArray χρησιμοποιεί τις βιβλιοθήκες iostream, fstream και string. Μέσα στο αρχείο header της κλάσης υπάρχει η κλάση OrderedObj η οποία χρησιμοποιείται για να υλοποιήσει τα στοιχεία του πίνακα. Η OrderedObj έχει ως ιδιότητες μία συμβολοσειρά word που είναι λέξη του στοιχείου, έναν ακέραιο occurrences που είναι ο αριθμός των εμφανίσεων της λέξης του στοιχείου. Ο κατασκευαστής της OrderedObj παίρνει ως όρισμα μία συμβολοσειρά την οποία την αναθέτει στην word και αρχικοποιεί την τιμή των occurrences με 1. Στο private κομμάτι της OrderedArray υπάρχει ο πίνακας OArray από στοιχεία OrderedObj, ο οποίος είναι ένας δυναμικός πίνακας από δείκτες σε αντικείμενα και αποτελεί την βάση της δομής. Επιπλέον υπάρχει η ιδιότητα size που είναι ένας

ακέραιος αριθμός που ορίζει πόσο μεγάλος μπορεί να είναι ο πίνακας , η ιδιότητα counter που είναι ένας ακέραιος αριθμός που μετράει πόσα στοιχεία είναι μέσα στον πίνακα OArray και η ιδιότητα sorted που είναι true/false αντίστοιχα με το αν είναι ο πίνακας ταξινομημένος. Οι μέθοδοι της κλάσης είναι:

-Κατασκευαστής: Ίδιος με την UnorderedArray
-Καταστροφέας: Ίδιος με την UnorderedArray
-void Insert(string): Ίδια με την UnorderedArray.
-bool Search(string): Η μέθοδος αυτή αναζητά στον πίνακα την λέξη που δίνεται ως όρισμα. Ελέγχει πρώτα άμα ο πίνακας είναι ταξινομημένος αλλιώς τον ταξινομεί και καλεί τον εαυτό της αναδρομικά. Έπειτα χρησιμοποιώντας δυαδική αναζήτηση ψάχνει την λέξη όρισμα στον πίνακα. Άμα την βρει επιστρέφει true αλλιώς επιστρέφει false.

-bool Delete(string): Η μέθοδος πρώτα ελέγχει άμα ο πίνακας είναι ταξινομημένος αλλιώς τον ταξινομεί και καλεί τον εαυτό της αναδρομικά. Καλεί την PosSearch για να βρει την θέση του αντικειμένου που αναζητά. Άμα δεν βρεθεί το αντικείμενο επιστρέφει false. Διαφορετικά διαγράφει το αντικείμενο και κάνει push όλα τα αντικείμενα που βρίσκονται στις επόμενες θέσεις κατά μια θέση για να καλυφθεί το κενό. Έπειτα μειώνει το counter κατά ένα και επιστρέφει true.

-bool PrintSearch(string): Η μέθοδος αυτή είναι ίδια με την παραπάνω Search, με την εξαίρεση ότι όταν βρίσκει την λέξη, εκτός από το να επιστρέφει true, την εκτυπώνει μαζί με τον αριθμό εμφανίσεων της στο αρχείο το οποίο δέχεται ως όρισμα.

-void QuickSort(): Η public έκδοση της QuickSort μπορεί να καλεστεί από την main. Ελέγχει αν ο πίνακας δεν είναι ταξινομημένος και αν δεν ειναι καλεί την private Quicksort με παραμέτρους την αρχή και το τέλος ολόκληρου του πίνακα και έπειτα θέτε το sorted ως true. Διαφορετικά εμφανίζει στην οθόνη ότι ο πίνακας είναι ήδη ταξινομημένος.
-int PosSearch(string): Η μέθοδος είναι ίδια με την Search με την διαφορά ότι επιστρέφει το index του στοιχείου που αναζητείται στο πίνακα. Αν δεν βρεθεί το στοιχείο επιστρέφει τον αριθμό -1.

-void QuickSort(int, int):Η private έκδοση της Quicksort καλείται μέσα από τις άλλες μεθόδους της κλάσης. Χρησιμοποιώντας έναν αλγόριθμο quicksort ταξινομεί τα στοιχεία από το πρώτο int index μέχρι το δεύτερο int index όρισμα. Λόγο της πρακτικής διαίρει και βασίλευε καλεί πολλές

φορές αναδρομικά τον εαυτό της. Επίσης κάνει χρήση των μεθόδων **void swap(int, int), int partition (int, int, int)** για την ταξινόμηση.

- **ΚΛΑΣΗ: BinarySearchTree:**
  Η BinarySearchTree χρησιμοποιεί τις βιβλιοθήκες iostream, fstream και string. Στο private κομμάτι της υπάρχει η κλάση Node, η οποία υλοποιεί τον κόμβο του δέντρου και έχει ως ιδιότητες μία συμβολοσειρά word που είναι λέξη του κόμβου, έναν ακέραιο occurrences που είναι ο αριθμός των εμφανίσεων της λέξης του κόμβου και δύο δείκτες right και left που δείχνουν στον δεξί και το αριστερό κόμβο αντίστοιχα. Ο κατασκευαστής της Node παίρνει ως όρισμα μία συμβολοσειρά την οποία την αναθέτει στην word και αρχικοποιεί τις τιμές των occurrences, right και left. Η Node έχει επίσης μία μέθοδο Print, η οποία εκτυπώνει τα περιεχόμενα του κόμβου σε ένα αρχείο. Στο private κομμάτι της BinarySearchTree υπάρχει και μία ιδιότητα, η root, που είναι ένας δείκτης στον κόμβο-ρίζα του δέντρου. Οι μέθοδοι της κλάσης είναι:

-Κατασκευαστής: Ο κατασκευαστής απλά αρχικοποιεί το root σε κενό δείκτη.

-Καταστροφέας-void Clear(Node*): Ο καταστροφέας καλεί την Clear, η οποία βρίσκεται στο private κομμάτι της κλάσης, γιατί χρησιμοποιείται μόνο εντός της, και πραγματοποιεί μία μεταδιατεταγμένη διάσχιση του δέντρου διαγράφοντας κάθε κόμβο που επισκέπτεται.

-void Insert(string): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης, καθώς χρησιμοποιείται από το πρόγραμμα. Η Insert δέχεται ως όρισμα μία συμβολοσειρά και αρχικά ελέγχει αν υπάρχει ο κόμβος-ρίζα και αν δεν υπάρχει εισάγει την νέα λέξη σε αυτόν. Αν υπάρχει τότε ψάχνει το δέντρο μέχρι να βρει έναν άδειο κόμβο για να εισάγει την νέα λέξη, εκτός αν υπάρχει ήδη η λέξη αυτή σε κάποιο κόμβο του δέντρου, στην οποία περίπτωση αυξάνει το occurrences του κόμβου αυτού κατά 1.

-bool Search(string): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης, καθώς χρησιμοποιείται από το πρόγραμμα. Η Search δέχεται ως όρισμα μία συμβολοσειρά και πραγματοποιεί μία τυπική αναζήτηση δυαδικού δέντρου αναζήτησης ψάχνοντας την συμβολοσειρά αυτή. Αν την βρει επιστρέφει true, ενώ αν κατά την αναζήτηση καταλήξει σε null δείκτη, που σημαίνει ότι η λέξη δεν υπάρχει στο δέντρο, επιστρέφει false.

-bool PrintSearch(string, ofstream&): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης και είναι ίδια με την παραπάνω Search, με την εξαίρεση ότι όταν βρίσκει την λέξη, εκτός από το να επιστρέφει true, την εκτυπώνει μαζί με τον αριθμό εμφανίσεων της στο αρχείο το οποίο δέχεται ως όρισμα.

-bool Delete(string): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης και χρησιμοποιείται από το πρόγραμμα για να διαγράψει τον κόμβο που περιέχει την λέξη που περνιέται ως όρισμα σε αυτήν, αν υπάρχει. Αρχικά, δηλώνονται δύο κενή δείκτες σε κόμβους, child και parent, και περνιούνται στην μέθοδο Search (στον $2^{ο}$ ορισμό της) με αναφορά. Αν η Search δεν βρει κάποιον κόμβο με την δοσμένη λέξη τότε η Delete επιστρέφει false. Αν βρεθεί ο κόμβος με την δοσμένη λέξη και ο γονέας του κόμβου αυτού, τότε καλείται η μέθοδος Delete($2^{ος}$ ορισμός της) , η οποία δέχεται τους δείκτες σε κόμβο child και τον γονέα του και αναλαμβάνει την αποδέσμευση του child από το δέντρο, και τέλος επιστρέφεται true.

-bool Search(string, Node*&, Node*&): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να βρεθεί ένας προς διαγραφή κόμβος και ο γονέας του. Είναι παρόμοια με τις παραπάνω Search, με την διαφορά ότι αν βρει τον κόμβο με την λέξη που αναζητά, εκτός από το να επιστρέφει true, αναθέτει στις μεταβλητές-δείκτες που δέχεται με αναφορά τους δείκτες στον προς διαγραφή κόμβο και τον γονέα του.

-void Delete(Node*, Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να διαγράψει έναν κόμβο από το δέντρο. Δέχεται ως ορίσματα δύο δείκτες σε κόμβους, τον child που είναι ο προς διαγραφή κόμβος και τον parent που είναι ο γονέας του child. Στην αρχή δηλώνεται μία bool μεταβλητή, η flag, για να ξέρουμε αν ο child είναι δεξί (true) ή αριστερό (false) παιδί του parent. Διακρίνονται 3 περιπτώσεις διαγραφής:

1. Ο child είναι φύλλο και δεν έχει παιδιά, δηλαδή ο left και ο right του είναι null δείκτες. Αν υπάρχει γονέας τότε θέτουμε τον δείκτη του parent που δείχνει στον child null και στην συνέχεια αποδεσμεύουμε τον child από την μνήμη. Αν δεν υπάρχει γονέας, τότε σημαίνει ότι διαγράφουμε την ρίζα του δέντρου.

2. Ο child είναι κόμβος με ένα μόνο υπόδεντρο. Έστω ότι ο child έχει μόνο δεξί υπόδεντρο. Αν υπάρχει γονέας, τότε στον δείκτη του γονέα που έδειχνε στον child αντιγράφουμε τον δείκτη που δείχνει στον δεξί κόμβο του child και στην συνέχεια αποδεσμεύουμε τον child από την μνήμη. Αν δεν υπάρχει γονέας απλώς διαγράφουμε την ρίζα και την αντικαθιστούμε με το δεξί της παιδί. Στην περίπτωση μόνο αριστερού υποδέντρου εργαζόμαστε ανάλογα.

3. Ο child είναι κόμβος με δύο υπόδεντρα. Εδώ δηλώνουμε τρεις καινούργιους δείκτες σε κόμβους, τους minRightNode, parentMinRightNode και τον temp. Καλούμε την συνάρτηση FindMinOfRight περνώντας με αναφορά τους minRightNode και parentMinRightNode, η οποία βρίσκει τον μικρότερο κόμβο (και τον γονέα του) του δεξιού υποδέντρου του child. Στην συνέχεια, δεσμεύουμε στον temp χώρο για έναν κόμβο και αντιγράφουμε σε αυτόν την λέξη και τις εμφανίσεις του minRightNode. Διαγράφουμε τον minRightNode από την αρχική του θέση στο δέντρο ξανακαλώντας την Delete, ωστόσο δεν υπάρχει κίνδυνος ατέρμονης επανάληψης, αφού ο minRightNode έχει το πολύ ένα δεξί υπόδεντρο και άρα θα πέσει σε μία από τις δύο παραπάνω περιπτώσεις διαγραφής. Στη συνέχεια βάζουμε τους right και left δείκτες του temp να δείχνουν στους αντίστοιχους right και left του child και αποδεσμεύουμε τον child από την μνήμη. Αν υπάρχει γονέας, τότε στον δείκτη του γονέα που έδειχνε στον child αντιγράφουμε τον temp, ενώ αν δεν υπάρχει σημαίνει πως διαγράψαμε την ρίζα, οπότε αντιγράφουμε τον temp στην ρίζα.

void FindMinOfRight(Node*, Node*&, Node*&): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της κατά την διαγραφή ενός κόμβου με δύο παιδιά. Δέχεται ως όρισμα έναν δείκτη στον προς διαγραφή κόμβο με δύο παιδιά και άλλους δύο δείκτες σε κόμβους με αναφορά. Αναζητεί τον αριστερότερο κόμβο του δεξιού υποδέντρου του πρώτου κόμβου που δέχτηκε ως όρισμα και μόλις τον βρει αναθέτει αυτόν και τον γονέα του στους δύο δείκτες που δέχτηκε με αναφορά.

void Preorder() / void Preorder(Node*, ofstream&)
void Inorder() / void Inorder(Node*, ofstream&)
void Postorder() / void Postorder(Node*, ofstream&) : Για κάθε τρόπο διάσχισης του δέντρου δημιουργήσαμε 2 μεθόδους στην κλάση, μια public που καλείται από το πρόγραμμα και μια private που την καλεί η

αντίστοιχη της public. Αυτό έγινε γιατί για την διάσχιση του δέντρου χρειάζεται να περνιέται ως όρισμα σε κάθε μέθοδο ένας δείκτης σε κόμβο, ενώ το πρόγραμμα δεν έχει πρόσβαση στην κλάση Node της BinarySearchTree. Όλες οι διασχίσεις εκτυπώνονται σε αρχεία κειμένου.

- **ΚΛΑΣΗ: AVLTree:**
  Η AVLTree χρησιμοποιεί τις βιβλιοθήκες iostream, fstream και string. Αποφασίσαμε η κλάση αυτή να μην κληρονομεί την BinarySearchTree, καθώς οι περισσότερες μέθοδοι έπρεπε να ξαναγραφούν για να δουλεύουν αναδρομικά και επειδή χρειαζόμαστε μια άλλη εκδοχή της κλάσης Node. Αυτή η εκδοχή της Node είναι ίδια με της BinarySearchTree, με επιπλέον μία ιδιότητα int height και δύο μεθόδους, την void UpdateHeight(), η οποία ρυθμίζει το ύψος του κόμβου ανάλογα με το πόσα παιδιά έχει, και την int GetBalance(), η οποία επιστρέφει την διαφορά του ύψους το δεξιού με του αριστερού υποδέντρου του κόμβου. Όπως η BinarySearchTree, η AVLTree έχει μία private ιδιότητα, την root, που είναι ο δείκτης στον κόμβο-ρίζα του δέντρου. Οι μέθοδοι της κλάσης είναι:

-Κατασκευαστής: Ίδιος με της BinarySearchTree..

-Καταστροφέας-void Clear(Node*): Ίδιος με της BinarySearchTree.

Void Preorder() / void Preorder(Node*, ofstream&) void Inorder() / void Inorder(Node*, ofstream&) void Postorder() / void Postorder(Node*, ofstream&) : Ίδιες με της BinarySearchTree.

-bool Search(string): Ίδια με της BinarySearchTree.

-bool PrintSearch(string, ofstream&): Ίδια με της BinarySearchTree.

void Insert(string): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης και χρησιμοποιείται για την εισαγωγή μιας λέξης στο δέντρο από το πρόγραμμα. Αρχικά ελέγχει αν υπάρχει η ρίζα και αν δεν υπάρχει εισάγει σε αυτήν την λέξη που δέχεται ως όρισμα. Αν υπάρχει καλεί έναν δεύτερο ορισμό της Insert, η οποία εισάγει τον κόμβο στο δέντρο και κρατάει αναδρομικά την ισορροπία του δέντρου.

void Insert(string, Node*, Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται εντός της για να εισαχθεί

ένας κόμβος στο δέντρο, ενώ διατηρείται η ισορροπία του. Δέχεται 3 ορίσματα, την λέξη που θα εισάγει και δύο δείκτες σε κόμβους τον currentNode και τον γονέα του currentNode. Η λογική της εισαγωγής είναι η ίδια με αυτήν του BinarySearchTree, απλώς η αναζήτηση γίνεται αναδρομικά αντί για επαναληπτικά. Στο τέλος της μεθόδου καλείται στον currentNode η UpdateHeight(), η οποία ενημερώνει το ύψος του, και στην συνέχεια περνιέται στην Balance(), η οποία ελέγχει αν ο κόμβος χρειάζεται περιστροφές. Με την χρήση της αναδρομής κάθε κλήση της Insert αντιστοιχεί σε έναν πρόγονο του κόμβου που θέλουμε να εισάγουμε, έτσι καταφέρνουμε να ελέγχουμε το ύψος όλων τον κόμβων που προσπελάθηκαν μέχρι να εισαχθεί ο νέος κόμβος.

void Balance(Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να ελέγξει αν ένας κόμβος χρειάζεται περιστροφές. Δέχεται ως όρισμα έναν δείκτη στον κόμβο τον οποίο θα ελέγξει και αρχίζει αποθηκεύοντας την ισορροπία του κόμβου σε μία μεταβλητή. Ανάλογα με την τιμή της ισορροπίας διακρίνονται δύο περιπτώσεις, με δύο υποπεριπτώσεις η καθεμία.

1. Ισορροπία = -2, δηλαδή το ύψος του αριστερού υποδέντρου είναι κατά 2 μεγαλύτερο από το ύψος του δεξιού υποδέντρου. Στην περίπτωση αυτήν ελέγχουμε και την ισορροπία του αριστερού κόμβου.

    A. αν η ισορροπία του αριστερού κόμβου είναι θετική, δηλαδή το δεξί υποδέντρο έχει μεγαλύτερο ύψος από το αριστερό, τότε περιστρέφουμε προς τα αριστερά τον αριστερό κόμβο και στην συνέχεια περιστρέφουμε προς τα δεξιά τον ίδιο τον κόμβο.

    B. στην άλλη περίπτωση απλώς περιστρέφουμε τον κόμβο προς τα δεξιά.

2. Ισορροπία = 2, δηλαδή το ύψος του δεξιού υποδέντρου είναι κατά 2 μεγαλύτερο από το ύψος του αριστερού υποδέντρου. Στην περίπτωση αυτή ελέγχουμε και την ισορροπία του δεξιού κόμβου.

    A. αν η ισορροπία του δεξιού κόμβου είναι 0 ή θετική απλώς περιστρέφουμε τον κόμβο προς τα αριστερά.

    B. στην άλλη περίπτωση, όπου το ύψος του αριστερού υποδέντρου είναι μεγαλύτερο του δεξιού, περιστρέφουμε προς τα δεξιά τον δεξιό κόμβο και στην συνέχεια περιστρέφουμε προς τα αριστερά τον ίδιο τον κόμβο.

void RotateRight(Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να περιστρέψει έναν κόμβο προς τα δεξιά. Δέχεται ως όρισμα έναν δείκτη x στον κόμβο στον οποίον θα πραγματοποιήσει περιστροφή και αρχικά θέτει τον δείκτη του αριστερού κόμβου του x ως y. Στην συνέχεια δημιουργούμε έναν νέο κόμβο τον tempxRight, ο οποίος θα είναι ο κόμβος που θα βρίσκεται δεξιά του x μετά την περιστροφή, και τον κατασκευάζουμε όπως θα πρέπει να είναι μετά την περιστροφή. Δημιουργούμε ακόμα έναν κόμβο τον tempx, ο οποίος θα είναι ο κόμβος που θα βρίσκεται στην θέση του x μετά την περιστροφή, και ξανά τον κατασκευάζουμε όπως θα πρέπει να είναι μετά την περιστροφή. Στην συνέχεια πραγματοποιούμε την περιστροφή συνδέοντας τους κόμβους που κατασκευάσαμε με το δέντρο και διαγράφουμε του κόμβους που περισσεύουν. Τέλος, καλούμε την UpdateHeight() στον κόμβο x και τον δεξιό κόμβο του x.

void RotateLeft(Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να περιστρέψει έναν κόμβο προς τα αριστερά. Η μέθοδος αυτή ακολουθεί ακριβώς την ίδια λογική με την RotateRight, με την διαφορά ότι τώρα περιστρέφει τον κόμβο προς αριστερά.

bool Delete(string): Η μέθοδος αυτή βρίσκεται στο public κομμάτι της κλάσης και χρησιμοποιείται από το πρόγραμμα για την διαγραφή ενός κόμβου που έχει μία δοσμένη λέξη. Δέχεται ως όρισμα την λέξη αυτή και αρχικά δηλώνεται μια bool μεταβλητή, η found. Στην συνέχεια η μεταβλητή αυτή περνιέται με αναφορά στον 2^{ο} ορισμό της συνάρτησης Delete, η οποία βρίσκει τον κόμβο με την δοσμένη λέξη και τον διαγράφει, ενώ διατηρεί την ισορροπία του δέντρου. Αν μετά την εκτέλεση της 2^{ης} Delete η found είναι true, αυτό σημαίνει ότι ο κόμβος βρέθηκε και διαγράφηκε οπότε επιστρέφουμε true, ενώ αν η found είναι false, αυτό σημαίνει ότι δεν υπάρχει κόμβος με την δοσμένη λέξη για να διαγραφεί, οπότε επιστρέφουμε false.

void Delete(string, Node*, Node*, bool&): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για να διαγράψει έναν κόμβο με την δοσμένη λέξη, ενώ διατηρεί την ισορροπία του δέντρου. Ψάχνει αναδρομικά το δέντρο μέχρι να βρει τον κόμβο με την δοσμένη λέξη και αν τον βρει τότε καλεί μία 3^{η} Delete, η οποία αναλαμβάνει την αποδέσμευση του κόμβου από το δέντρο, και κάνει την found true. Αν δεν την βρει κάνει την found false. Στο τέλος της

μεθόδου καλείται στον κόμβο η UpdateHeight() και έπειτα περνιέται στην Balance(), για να διατηρείται η ισορροπία του δέντρου.

void Delete(Node*, Node*): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για την αποδέσμευση ενός κόμβου από το δέντρο. Είναι ίδια με την αντίστοιχη Delete της BinarySearchTree με την μόνη διαφορά να βρίσκεται στην 3η περίπτωση διαγραφής. Η λογική της διαγράφης ενός κόμβου με δύο παιδιά είναι ακριβώς η ίδια με αυτήν της BinarySearchTree, η μόνη διαφορά είναι πως χρησιμοποιούμε μια τροποποιημένη FindMinOfRight(), η οποία αντί να επιστρέφει με αναφορά έναν δείκτη στον μικρότερο κόμβο του δεξιού υποδέντρου, επιστρέφει με αναφορά την λέξη και τις εμφανίσεις του κόμβου αυτού, αφότου τον έχει διαγράψει.

void FindMinOfRight(Node*, Node*, string&, int&): Η μέθοδος αυτή βρίσκεται στο private κομμάτι της κλάσης, καθώς χρησιμοποιείται μόνο εντός της για την διαγραφή ενός κόμβου με δύο παιδιά. Σε αντίθεση με την αντίστοιχη μέθοδο της BinarySearchTree δέχεται ως πρώτο όρισμα το δεξί παιδί του κόμβου που θέλουμε να διαγράψουμε και όχι τον ίδιο τον κόμβο. Ψάχνει αναδρομικά να βρει τον αριστερότερο κόμβο και μόλις τον βρει αναθέτεις τα περιεχόμενα του στις μεταβλητές που δέχτηκε με αναφορά και καλεί την 2η Delete σε αυτόν. Στο τέλος της μεθόδου καλείται στον κόμβο η UpdateHeight() και έπειτα περνιέται στην Balance(), για να διατηρείται η ισορροπία του δέντρου.

- **ΚΛΑΣΗ:HashTable:**

  Η HashTable χρησιμοποιεί τις βιβλιοθήκες iostream, fstream και string. Ο συγκεκριμένος πίνακας κατακερματισμού έχει υλοποιηθεί με την τεχνική ανοιχτής διεύθυνσης (open adressing) και με την γραμμική δοκιμή (linear probing). Μάλιστα, έχει δημιουργηθεί και ένα αντικείμενο της κλάσης HashTable (το HashTableObject) για την υλοποίηση της. Το αντικείμενο αυτό περιλαμβάνει έναν κενό constructor και 2 μεταβλητές. Οι μέθοδοι της κλάσης HashTable ειναι:

Καστασκευαστής: Ο κατασκευαστής θέτει στην μεταβλητή size το μέγεθος του πίνακα της κλάσης, αρχικοποιεί την μεταβλητή counter και δημιουργεί έναν νέο πίνακα κατασκευασμένο από δείκτες, οι οποίοι

δείχνουν στο αντικείμενο της κλάσης (HashTableObject).Επίσης οι συγκεκριμένοι δείκτες αρχικοποιούνται με NULL.

Καταστροφέας: Ο καταστροφέας διαγράφει την συγκεκριμένη θέση του πίνακα και το περιεχόμενο του όταν δεν είναι NULL. Στο τέλος του προγράμματος διαγράφει ολοκληρωτικά τον πίνακα, δηλαδή τις κενές θέσεις.

int Hash_function(string):Η συνάρτηση κατακερματισμού δέχεται ως όρισμα μια λέξη και επιστρέφει τον δείκτη (index),ο οποίος δείχνει σε μια θέση στον πίνακα .Ακόμα περιέχει την μεταβλητή sum, η οποία ειναι unsigned long long επειδή υπάρχει η πιθανότητα το άθροισμα που προκύπτει από την παρακάτω for να είναι πολύ μεγάλος αριθμός. Ο index προκύπτει παίρνοντας το υπόλοιπο της διαίρεσης του αθροίσματος με το μεγέθος του πίνακα.

void Insert(string):Η μέθοδος αυτή δεν επιστρέφει τίποτα όταν γεμίσει ο πίνακας κατακερματισμού. Όταν ο πίνακας έχει κενές θέσεις (φαίνονται από το nullptr), με βάση την συνάρτηση κατακερματισμού, βρίσκει την σωστή θέση που πρέπει να τοποθετηθεί η συγκεκριμένη λέξη. Σε περίπτωση που υπάρχει σύγκρουση (collision) με άλλη λέξη, λόγω ίδιου index, τότε πηγαίνει στην επόμενη θέση και αν πάλι βρεθεί διαφορετική λέξη στην συγκεκριμένη θέση πηγαίνει στην αμέσως επόμενη, δηλαδή δύο θέσεις πιο μετά από την αρχική που είχε βρει. Τέλος, το μοτίβο επαναλαμβάνεται μέχρις ότου να βρεθεί μια κενή θέση. Στην περίπτωση που βρεθεί η ίδια λέξη θα αυξηθεί η μεταβλητή occurrences κατά ένα. Μάλιστα, μετά από κάθε λέξη ενημερώνονται οι δείκτες.

bool Search(string):Η μέθοδος αυτή ακολουθεί την ίδια μεθοδολογία με την insert έτσι ώστε να μπορεί να εντοπίσει τις λέξεις που έχουν εισαχθεί. Μάλιστα, σε περίπτωση που εντοπίσει κάποια λέξη επιστέφει true, αλλιώς επιστρέφει false.

bool PrintSearch(string, ofstream&): Η μέθοδος αυτή είναι ίδια με την παραπάνω Search, με την εξαίρεση ότι όταν βρίσκει την λέξη, εκτός από το να επιστρέφει true, την εκτυπώνει μαζί με τον αριθμό εμφανίσεων της στο αρχείο το οποίο δέχεται ως όρισμα.