Δομές δεδομένων Εργασία Δεύτερη

Πλήρης αναλυτική παρουσίαση όλων των μεθόδων και κλάσεων της εργασίας.

Σκοπός αυτής της εργασίας είναι η δημιουργία ενός αγγλοελληνικού λεξικού με την βοήθεια ενός δυαδικού δέντρου αναζήτησης(Binary Search Tree).Η δημιουργία του προγράμματος και η υλοποίηση των βασικών λειτουργιών του λεξικού βασίστηκε στις αρχές του αντικειμενοστραφούς προγραμματισμού και στην γλώσσα προγραμματισμού C++.

Παρουσίαση των βασικών επιμέρους κλάσεων του προγράμματος.

1.Κλάση Node

Σε αυτό το header file ορίζουμε την κλάση Node ο οποίος θα αποτελέσει τον κόμβο του Binary Search Tree που θα δημιουργήσουμε .Ο κάθε κόμβος αποτελείται από τα εξής πεδία : greek (λέξη στα ελληνικά), english (λέξη στα αγγλικά) και τα πεδία leftChild και rightChild που είναι και δείκτες τύπου Node στο αριστερό και δεξί παιδί αντίστοιχα του δένδρου που θα φτιάξουμε. Επίσης δηλώνουμε ότι η κλάση Node είναι φίλη κλάση της BST οπότε η BST θα μπορεί να έχει πρόσβαση στα ιδιωτικά μέλη της Node.Επίσης ορίζουμε στα public μέλη διαφορετικούς constructor (overloading) που έχουν διαφορετικό πλήθος ορισμάτων.

Στο source file αρχείο παρέχονται διαφορετικοί constructors με διαφορετικά ορίσματα .Ο πρώτος δεν δέχεται ορίσματα και δίνει αρχικές τιμές στα πεδία leftChild και rightChild με 0(δηλαδή με Null).Ο δεύτερος constructor δέχεται ορίσματα τα string e1 και e2 και αποδίδει αρχικές τιμές στα πεδία english και greek και στα πεδία leftChild και rightChild με Null. Ο τρίτος constructor δέχεται ορίσματα τα string e1 και e2 και pointers *l

και *r αποδίδει αρχικές τιμές στα πεδία english και greek και στα πεδία leftChild και rightChild με τις διευθύνσεις που δείχνουν οι προηγούμενοι Pointers.

2.Κλάση LinkedList

Στο header file ορίζουμε την κλάση LinkedList .Αυτή περιλαμβάνει στα δημόσια μέλη την int μεταβλητή length,τον pointer *head ,τον constructor LinkedList(),τον αποδομητή ~LinkedList() ,την συνάρτηση void add(string data1,string data 2), την συνάρτηση void print() και τέλος την συνάρτηση search(string word)αναζήτησης.

Στο source file ορίζουμε τον constructor LinkedList() η οποία αρχικοποιεί τα πεδία length και head με 0. Ο deconstructor ~LinkedList() διαγράφει όλους του κόμβους της διασυνδεδεμένης αλυσίδας. Ορίζουμε έναν pointer τύπου LNode *next.Όσο το head δείχνει σε κάποιον κόμβο(δηλαδή έχει ως τιμή την διεύθυνση κάποιου άλλου κόμβου) και δεν είναι Null ο next παίρνει σαν τιμή την διεύθυνση του επόμενου δείκτη που δείχνει ο head,διαγράφουμε τον κόμβο που δείχνει ο head και ο head λαμβάνει την διεύθυνση που δείχνει ο next δηλαδή στον επόμενο κόμβο .Τέλος εμφανίζεται μήνυμα η λίστα ότι διαγράφτηκε.

Η add(string data1,string data2) μέθοδος της linked list υλοποιεί την βασική ιδέα του αλγορίθμου insertion sort.Δημιουργούμε έναν κόμβο newNode τύπου LNode Δημιουργούμε δύο pointers τύπου LNode *p,*f ο οποίος είναι ίσος με τον δείκτη head (δείχνουν στην ίδια διεύθυνση). Θέτουμε το πεδίο data1 του newNode και το πεδίο data2 του newNode με τιμές data1 και data2.Το πεδίο του link του newNode ίσο με null και αυξάνουμε το μέγεθος της λίστας. Αν το head ίσο με null τότε σημαίνει ότι η λίστα είναι άδεια και ο κόμβος που θα προστεθεί να ο πρώτος της αλυσίδας. Τότε το

πεδίο link θα ισούται με τον δείκτη Null και το head θα ισούται με την διεύθυνση που δείχνει ο κόμβο newNode.Αν δεν ισχύει αυτή η συνθήκη τότε εξετάζεται η περίπτωση τα δεδομένα του data1 του ορίσματος που είναι ουσιαστικά η λέξη στα αγγλικά είναι μικρότερη από την αντίστοιχη λέξη του head .Αν είναι αληθής τότε το πεδίο link του newNode γίνεται ίσο με το head (δείχνουν στην ίδια διεύθυνση) και ο δείκτης Head θα δείχνει στο newNode .Αν είναι ψευδής η συνθήκη εξετάζεται η περίπτωση το newNode να εισαχθεί ενδιάμεσα στην αλυσίδα είτε στο τέλος αυτής. Όσο ο δείκτης f είναι διαφορετικός του null ελέγχουμε δυο συνθήκες 1)αν τα δεδομένα data1 είναι μικρότερα από αυτά του f->data1(λέξεις στα αγγλικά) ο δείκτης p να πάρει την διεύθυνση που έχει ως τιμή ο f και ο f να είναι ίσος με την διεύθυνση του πεδίου του link και μετά χρησιμοποιούμε την εντολή continue για να προσπεράσουμε αυτήν την επανάληψη.(Δεν μπορεί να εισαχθεί σε αυτό το σημείο ο κόμβο ώστε να διατηρηθεί η ταξινόμηση) 2)Αν δεν ισχύει αυτή η συνθήκη 1 τότε το πεδίο p link θα δείχνει στον κόμβο του newNode και το πεδίο link του newNode τίθεται ίσο με την διεύθυνση που δείχνει ο f.Τέλος αν η λίστα είναι γεμάτη τότε προσθέτουμε τον κόμβο στο τέλος της αλυσίδας.

Η μέθοδος void print() εκτυπώνει τα πεδία data1 δηλαδή της λέξεις που είναι στα αγγλικά. Δημιουργούμε έναν pointer *head τύπου LNode.Όσο ο δείκτης head δεν είναι Null δηλαδή δείχνει κάπου εμφανίζουμε τα δεδομένα του κόμβου που δείχνει ο head και στην συνέχεια βάζουμε το head δείχνει στον επόμενο κόμβο.

Η μέθοδος search(string word) πραγματοποιεί αναζήτηση στην διασυνδεδεμένη λίστα . Αρχικά ορίζουμε έναν δείκτη τύπου LNode *current .Το current δείχνει εκεί που δείχνει ο head .Όσο current δεν είναι null και το πεδίο data1 (δηλαδή ή λέξη στα αγγλικά δεν είναι ίση με την λέξη word1) βάζουμε το

current να δείχνει στους επόμενους κόμβους. Στο τέλος του βρόγχου while αν τα δεδομένα του current και συγκεκριμένα το πεδίο data1 είναι ίσο με την λέξη word εμφανίζεται η αντίστοιχη λέξη αλλιώς εμφανίζεται μήνυμα σφάλματος.

3.Κλάση LNode

Στο header file αρχείο ορίζουμε στα δημόσια μέλη της έναν pointer *link του LNode και τα πεδία string data1,data2.

4.Κλάση ChainNode

Η μόνη διαφορά από την αντίστοιχη LNode είναι ότι περιλαμβάνει μόνο έναν πεδίο και μάλιστα τύπου χαρακτήρα.

5.Κλάση Chain

Αυτή περιλαμβάνει στα δημόσια μέλη την int μεταβλητή length,τον pointer *first,τον constructor Chain() ,την συνάρτηση void add(char e), την συνάρτηση int find(char e) ,την συνάρτηση void all() και τέλος την συνάρτηση int count().

Στο source file ορίζουμε έναν constructor Chain που αρχικοποιεί το length και το first. Η add(char e) δημιουργεί έναν κόμβο y.Τα δεδομένα στο y κόμβο και το πεδίο data1 και το θέτουμε ίσο με το e.Αν η λίστα είναι κενή δηλαδή η first είναι null το πεδίο link του y δείχνει εκεί που δείχνει ο first και ο δείκτης first λαμβάνει την διεύθυνση του y.Αν η λίστα δεν είναι κενή τότε όσο ο επόμενος κόμβος του k δείχνει κάπου δηλαδή δεν είναι null μέσα στο loop βάλε τον κ να δείχνει εκεί που δείχνει ο k->link.Θέτουμε το y->link ίσο με Null(αφού είναι ο τελευταίος κόμβος που προσθέτουμε δεν θα έχει επόμενο) και το k->link εκεί που δείχνει το y.

Η void all() εκτυπώνει όλα τα δεδομένα της λίστας. Δημιουργούμε έναν pointer *current τύπου Chain που αρχικά δείχνει στο first(δηλαδή εκεί που δείχνει και ο first).Όσο ο δείκτης head δεν είναι Null δηλαδή δείχνει κάπου

εμφανίζουμε τα δεδομένα του κόμβου που έχει ο current και στην συνέχεια βάζουμε το current δείχνει στον επόμενο κόμβο.

Η int count() επιστρέφει έναν μετρητή που περιέχει το πλήθος των στοιχείων της λίστας. Ορίζουμε έναν δείκτη τύπου ChainNode *current ο οποίος έχει την διεύθυνση του κρατά first(εκεί που δείχνει ο first)Επιπλέον ορίζουμε μια ακέρια μεταβλητή count.Όσο το current δείχνει κάπου (δηλαδή δεν είναι null)αύξησε την μετρητή count και counter κρατά την διεύθυνση που έχει το πεδίο current link δηλαδή του επόμενου κόμβου. Τέλος μόλις ολοκληρωθεί το loop επιστρέφει την ακέραια τιμή count.

Τέλος ορίζουμε την συνάρτηση find(char e).Ορίζουμε έναν δείκτη current τύπου ChainNode ο οποίος δείχνει εκεί που δείχνει και ο first.Ορίζουμε μια ακέραια μεταβλητή index και την θέτουμε ίση με ένα. Όσο το current δείχνει κάπου η και ο χαρακτήρας του κόμβου που δείχνει ο current δεν είναι ίσο με τον χαρακτήρα όρισμα βάζουμε τον current να δείχνει στο επόμενο κόμβο της αλυσίδας και αυξάνουμε το Index.Αν ο current δεν είναι null και έχουμε βγει από το while loop προφανώς σημαίνει ότι βρήκαμε το στοιχείο και επιστρέφουμε το index.

6.Η κλάση BST(Binary Search Tree)

Στο header file αρχείο ορίζουμε τα ιδιωτικά μέλη και τα δημόσια μέλη της κλάσης. Αρχικά στα ιδιωτικά ορίζουμε τον δείκτη *root που είναι τύπου Node(και συνήθως θα δείχνει προς τον κόμβο του δέντρου που θα αποτελεί ρίζα).Στα δημόσια μέλη ορίζουμε τον constructor BST(),την void συνάρτηση Delete(string e1),την συνάρτηση Search() η οποία είναι τύπου LinkedList και επιστρέφει έναν Pointer τέτοιου τύπου, την συνάρτηση findMin(Node *t)που επιστρέφει έναν

pointer τύπου Node*,την void συνάρτηση inorder(Node*t),την συνάρτηση inorder2(Node *t) η οποία επιστρέφει έναν pointer τύπου Chain,την void συνάρτηση Display(),την συνάρτηση deleteNode(Node *root,string data) η οποία επιστρέφει έναν pointer τύπου Node ,την συνάρτηση void Insert(string e1,string e2),την συνάρτηση insert(Node* leaf **,** string key **,**string key2) η οποία επιστρέφει έναν δείκτη τύπου Node, η συνάρτηση void print(Node *p,int depth),η συνάρτηση void levelorder() και η συνάρτηση int depth(Node *p,int d).Τέλος η BST είναι φίλη κλάση της Node δηλαδή η Node έχει πρόσβαση στα ιδιωτικά μέλη της BST.

Στο source file αρχείο θα αναλύσουμε τις βασικές συναρτήσεις που ορίσαμε στο header αρχείο. Η insert(Node *leaf**,** string key**,** string key2) πραγματοποίει εισαγωγή στο δένδρο BST και συμβάλει στην δημιουργία του. Αρχικά εξετάζουμε αν ο κόμβος του δένδρου(ή και όλο το δένδρο) είναι κενό, δηλαδή το leaf είναι ίσο με null και αν ισχύει ,δημιουργούμε με δυναμική δέσμευση μνήμης ένα καινούργιο κόμβο που δείχνει ο leaf.Εκχωρούμαι τιμές στα πεδία English,Greek,rightChild ,leftChild ίσο με key,key2,null,null αντίστοιχα. Αν δεν είναι αληθής η συνθήκη εξετάζουμε την περίπτωση το string key να είναι μικρότερο από το leaf->english(συγκρίνουμε τις λέξεις στα αγγλικά)και αν όντως ισχύει ,πραγματοποιούμαι αναδρομή insert(leaf->leftChild,key,key2) και εκχωρούμαι το αποτέλεσμα στο leaf->leftChild (εκεί θα δείχνει το leaf->leftChild).Αλλιώς αντίστοιχα αν το key είναι μεγαλύτερο από το leaf->English πρέπει να κατευθυνθούμε για να κάνουμε εισαγωγή στο δεξιό υποδένδρο και εκχωρούμε στο πεδίο του rightChild leaf να δείχνει στο αποτέλεσμα της αναδρομής του επόμενου δεξιού κόμβου του υποδένδρου της αναδρομής insert(leaf->rightChild,key,key2).Μετά από τις συνθήκες επιλογής επιστρέφεται η διεύθυνση που κράτα ο pointer leaf.

Στην συνέχεια ορίζεται η συνάρτηση Insert(string e1,string e2) οπού αποθηκεύονται οι αλλαγές που πραγματοποιήθηκαν στο δένδρο μετά την εισαγωγή του νέου κόμβου(αλλαγές στην δομή του δένδρου)

Η συνάρτηση deleteNode(Node *p ,string data).Αν το p είναι null τότε επιστρέφει η διεύθυνση της root .Αν τα δεδομένα data1 είναι μικρότερα p->english τότε κινούμαστε στο αριστερό υποδένδρο και το p->leftChild θα είναι ίσο με την διεύθυνση deleteNode(p->leftChild, data).Αλλιώς αν τα δεδομένα data είναι μεγαλύτερα από την λέξη στα αγγλικά του κόμβου p τότε το πεδίο rightChild του p θα δείχνει στο αποτέλεσμα της αναδρομής deleteNode(p->rightChild,data) πηγαίνουμε στον δεξιό κόμβο του υποδένδρου p.Έπειτα ελέγχουμε τις εξής περιπτώσεις.1)Και το αριστερό δένδρο του p και το δεξί είναι null οπότε ο p φύλλο και επομένως διαγράφεται.2)Αν το πεδίο p του leftChild είναι ίσο με null ,ορίζουμε pointer τύπου Node*temp να δείχνει εκεί που δείχνει ο p και βάζουμε τον p να δείχνει εκεί που δείχνει το πεδίο του rightChild και διαγράφουμε αυτόν τον κόμβο.3) Αν το πεδίο p του rightChild είναι ίσο με null ,ορίζουμε pointer τύπου Node*temp να δείχνει εκεί που δείχνει ο p και βάζουμε τον p να δείχνει εκεί που δείχνει το πεδίο του leftChild και διαγράφουμε αυτόν τον κόμβο 4)Ορίζουμε έναν δείκτη τύπου Node *min ο οποίος έχει σαν τιμή εκεί που δείχνει το δεύτερο μέλος της ισότητας χρησιμοποιεί την συνάρτηση findMin(p->rightChild) βρίσκοντας το μικρότερο στοιχείο του δεξιού υποδένδρου του p.Το πεδίο english του p γίνεται ίσο πεδίο english του min και το p rightChild γίνεται δείχνει εκεί που δείχνει το αποτέλεσμα της αναδρομής deleteNode(p->rightChild, min->english).Τέλος επιστρέφεται η διεύθυνση που κρατά ο p.

Η συνάρτηση findMin(Node *p) λαμβάνει έναν δείκτη και όσο το αριστερό της παιδί δεν είναι null τότε αλλάζουμε το p ώστε να δείχνει στα αριστερά παιδία του. Έτσι επιτυγχάνεται να βρούμε το ελάχιστο στοιχείο συμφώνα με τις γνώσεις μας στα δυαδικά δένδρα αναζήτησης. Επιστρέφεται η διεύθυνση που κρατά ο δείκτης p.Η συνάρτηση Delete(string e1)διαγράφει από το δένδρο που ξεκινάει από την ρίζα το δεδομένο αν υπάρχει. Οι αλλαγές αποθηκεύονται στο root.

Η συνάρτηση Search()

Αρχικά ζητείται από τον χρήστη να πληκτρολογήσει τον αριθμό των αρχικών γραμμάτων που βάση αυτών θα πραγματοποιηθεί η αναζήτηση .Γίνεται έλεγχος ότι θα δοθεί ακέραιος αριθμός. Έπειτα ο χρήστης πληκτρολογεί μια λέξη και με βάση αυτή δημιουργούμε ένα πίνακα char μεγέθους n (αριθμός γραμμάτων που δόθηκε πριν) όπου τα στοιχεία του πίνακα θα αποτελούν τα n γράμματα της λέξης .Χρησιμοποιείται η εντολή strncpy(words, beginning.c_str(), n) η οποία αντιγράφει τα n γράμματα από το string beginning στον πίνακα words.Ορίζουμε έναν pointer *p που δείχνει εκεί που δείχνει ο root.Επίσης δημιουργούμε μια λίστα l (LinkedList).Όσο το p δεν είναι Null ελέγχουμε αν η λέξη του δυαδικού δένδρου στα αγγλικά είναι ίση στα πρώτα n γράμματα της .Αν αυτό όντως ισχύει προστίθεται η λέξη στην λίστα και η αντίστοιχη λέξη στα ελληνικά και εμφανίζεται μήνυμα επιτυχίας. Έπειτα γίνεται σύγκριση της λέξης στα αγγλικά με το string word(μετατροπή του πίνακα char σε string)μέσω της συνάρτησης compare.Αν η compare είναι θετική σημαίνει ότι πρέπει να κατευθυνθούμε στο δεξί υποδένδρο και ο δείκτης p να δείξει στο δεξί παιδί αλλιώς κατευθυνόμαστε στο αριστερό υπόδενδρο και ο δείκτης p δείχνει στο αριστερό παιδί. Επιστρέφεται η λίστα.

Η συνάρτηση inorder(Node *t) πραγματοποιεί ενδοδιαταγμένη διάσχιση του δένδρου. Αν ο δείκτης t είναι null σημαίνει ότι το δένδρο είτε είναι κενό είτε έχουμε φτάσει σε έναν κόμβο που είναι κενός οπότε τερματίζεται η συγκεκριμένη κλήση της συνάρτησης. Αλλιώς πραγματοποιείται αναδρομή στο αριστερό υποδένδρο της. Αργότερα μόλις φτάσει σε null το αριστερότερο υποδένδρο της εμφανίζονται οι λέξεις στα ελληνικά και στα αγγλικά και έπειτα πραγματοποιείται αναδρομή inorder στο δεξιό υποδένδρο. Η Display() καλεί την inorder που κάνει ενδοδιάταξη με όρισμα την root.

Για την λειτουργία της εκτύπωσης του δένδρου ως την ζητούμενη μορφή θα πρέπει να επισημανθούν τα εξής. Αρχικά παρατηρούμε ότι η εκτύπωση του δένδρου ανά γραμμή πραγματοποιείται σύμφωνα με την διάσχιση του δένδρου κατά σειρά επιπέδων(level order).Τα πλήθος των κενών όμως που εκτυπώνεται πριν από κάθε πρώτο γράμμα της αντίστοιχης λέξης στα αγγλικά καθορίζεται από τον πλήθος των στοιχείων που προηγούνται πριν από αυτό στην διάσχιση σύμφωνα με την ενδοδιάταξη. Για την υλοποίηση αυτού του ζητούμενος θα πρέπει να ορίσουμε την διαδικασία της ενδοδιάταξης χωρίς αναδρομή.

Η συνάρτηση inorder2(Node *t)

Ορίζουμε μια στοίβα τύπου Node*s και έναν δείκτη Node* curr που δείχνει εκεί που δείχνει και t.Επίσης φτιάχνουμε μια αλύσιδα Chain P αφού δεν γνωρίζουμε ακριβώς το πλήθος των στοιχείων που θα εισαχθούν σε αυτόν .Όσο ο δείκτης δεν είναι null ή στοίβα δεν είναι άδεια τότε εισερχόμαστε σε ένα άλλο loop όπου όσο ο δείκτης cur δεν είναι null ,τοποθέτησε τον τρέχοντα κόμβο στην στοίβα και βάλε τον curr να δείχνει στον αριστερό παιδί του. Όταν ο curr γίνει null τοποθέτησε τον κόμβο που βρίσκεται στην κορυφή στο curr ,κάνε εξαγωγή από την στοίβα και πραγματοποίησε την λειτουργία add του

πρώτου γράμματος της αντίστοιχης λέξης στα αγγλικά που βρίσκεται εκεί που δείχνει ο δείκτης curr.Βάζουμε το curr να δείξει στο δεξιό παιδί και εισερχόμαστε σε μια καινούργια επανάληψη του loop .Όταν η στοίβα αδειάσει τότε επιστρέφουμε την διασυνδεδεμένη λίστα P.

Το depth(Node *p, int d) .Το depth ενός δένδρου είναι το μέγιστο μονοπάτι από την ρίζα στα φύλλα(το ύψος του δένδρου). Η Base condition της αλγορίθμου της αναδρομής που εφαρμόζεται και στο depth(Node *p,int d) είναι αν ο δείκτης p είναι ίσος με null τότε επιστρέφεται ο Integer d.Στην συνέχεια πραγματοποιούνται ανάδρομες ώστε να υπολογίσουμε τις integer μεταβλητές left,right και χρησιμοποιώντας την συνάρτηση max επιλέγουμε το μέγιστο ύψος του δένδρου.

Η συνάρτηση print(Node *p, int depth).Αρχικά εξετάζουμε αν ο δείκτης p είναι null και αν ισχύει τερματίζεται η μέθοδος χωρίς να επιστρέφεται κάτι. Στην συνέχεια δημιουργούμε μια ουρά q τύπου Node* στην οποία προσθέτουμε τον p(συνήθως το όρισμα αυτό θα αντιπροσωπεύει την ρίζα).Χρησιμοποιώντας την inorder2 επιστρέφεται η λίστα chain και τίθεται στην λίστα Chain *c.Εκτυπώνουμε τα στοιχεία της που είναι αριθμημένα σύμφωνα με την ενδοδιάταξη και ορίζουμε έναν μετρητή που περιέχει το πλήθος των στοιχείων χρησιμοποιώντας την αντίστοιχη μέθοδο της Chain.Όσο η ουρά δεν είναι κενή ορίζουμε μια ακέραια μεταβλητή nodeCount που θα είναι ίση με το μέγεθος της λίστας και αν είναι κενή πραγματοποιείται έξοδος από το loop.Ορίζουμε ένα δεύτερο while loop και όσο η μεταβλητή nodeCount είναι μεγαλύτερη του μηδενός .Βάζουμε τον δείκτη *node να δείχνει στον πρώτο στοιχείο της ουράς .Πραγματοποιούμε αναζήτηση find της Chain και επιστρέφει την θέση που βρίσκεται το στοιχείο(λέξη στα αγγλικά πρώτο γράμμα) του δείκτη Node και εκτυπώνονται a-prev-1 “X”

κενά(Το prev ορίστηκε πιο πάνω με 0).Μετά εκτυπώνεται το πρώτο γράμμα της λέξης στα αγγλικά και θέτουμε το prev ίσο με την μεταβλητή a.Αν τα πεδία leftChild,rightChild δεν είναι null τα τοποθετούμε στην ουρά αντίστοιχα τα node->leftChild,node->rightChild και μειώνουμε το nodeCount κατά 1.Μέτα τον εμφωλευμένο βρόγχο εκτυπώνονται count-a “X” ώστε να διατηρείται η ανάλογη ομοιομορφία στις γραμμές. Μειώνουμε το depth κατά ένα και αλλάζουμε γραμμή εκτύπωσης των δεδομένων. Τέλος ορίζουμε την συνάρτηση levelOrder η καλεί την depth(root,1) οποία βρίσκει το depth του δένδρου και το χρησιμοποίει στο print(root,d).

7)Η main

Αρχικά ορίζουμε μια συνάρτηση int input(string a) η οποία διασφαλίζει ότι δεν θα δοθεί από τον χρηστή string

(κάτι που θα οδηγούσε σε compile error) ώστε να δοθεί ακέραιος αριθμός. Χρησιμοποιώντας εντολές cin.fail() ώστε να διασφαλίσθει ακέραιος αριθμός,cin.clear() για να καθαρίσει το ρεύμα εισόδου από ανεπιθύμητες εισόδους καθώς και cin.ignore(INT_MAX, '\n') ως όριο επιτυγχάνεται η επιστροφή ενός ακέραιου αριθμού.

Δημιουργούμε ένα BST με όνομα dictionary,έναν pointer l τύπου LinkedList και ορίζουμε string word1 ,word2 ,sW ,sav1 , sav2 ,ws.

Στην main δημιουργούμε μέσω του do-while loop και switch ένα μενού επιλογής. Αρχικά παρουσιάζονται όλες οι επιλογές και έπειτα ζητείται μέσω του χρήστη μέσω του input(string a)να δώσει έναν αριθμό από 1-5 όπως και οι διαθέσιμες επιλογές διασφαλίζοντας κάτι τέτοιο. Μέσω του switch statement ελέγχονται οι περιπτώσεις(Επίσης αναφέρεται ότι για όλα τα cases πραγματοποιείται break μετά την εκτέλεση των παραπάνω εντολών) .Αν δοθεί 1 ζητούνται δύο λέξεις μια

στα αγγλικά και μια στα ελληνικά και πραγματοποιούμαι insert στο binary search tree σύμφωνα με την αντίστοιχη μέθοδο .Αν δοθεί 2 ζητείται από τον χρήστη να δοθεί η λέξη που θα διαγραφτεί και καλούμαι την αντίστοιχη μέθοδο delete του BST.Αν δοθεί 3 pointer *l LinkedList που ορίσαμε πιο πάνω και βάζουμε να δείχνει στα στοιχεία της λίστας που επιστρέφεται μετά από την κλήση της Search.Καλούμε την συνάρτηση print της LinkedList για την εκτύπωση των στοιχείων της λίστας. Επίσης ζητείται από τον χρήστη να πληκτρολογήσει μια λέξη από τις παραπάνω που εμφανίστηκαν σε αύξουσα σειρά και πραγματοποιείται αναζήτηση από την search της LinkedList έτσι ώστε να εκτυπωθεί η αντίστοιχη λέξη στα ελληνικά.Αν δοθεί 4 πραγματοποιείται ενδοδιάταξη και εκτύπωση του δένδρου σύμφωνα με την Display() και αν τέλος δοθεί 5 καλείται η levelorder() συνάρτηση της BST κλάσης για να παρουσιαστεί το δένδρο ως δομή. Τέλος ζητείται από τον χρήστη να δώσει 0 αν θέλει να συνεχίσει να πραγματοποιήσει κάποια από τις παραπάνω λειτουργίες του λεξικού αλλιώς 1 για να τερματίσει το πρόγραμμα. Είναι τέλος ιδιαίτερα σημαντικό να τονισθεί ότι γίνεται σύμφωνα με την μέθοδο input(string a)και ένα επιπρόσθετο do-while ότι θα δοθεί ακέραιος αριθμός είτε 1 είτε 0.

Δημητρίου Δημήτρης Π18036