**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ**

**Σχολή: Τμήμα Πληροφορικής**

**Μάθημα: Δομές Δεδομένων**

**Όνομα: Παναγιωτάτος Γεώργιος - Καρολίδης Θεόδωρος**

**ΑΕΜ:**

* **Παναγιωτάτος Γεώργιος (2627)**
* **Καρολίδης Θεόδωρος (2572)**

**Project: AVL Tree**

Το πρόγραμμα που δημιουργήσαμε για το AVL tree λειτουργεί πλήρως.

Οι βιβλιοθήκες που χρησιμοποιήθηκαν στο πρόγραμμα είναι οι εξής :

1. <iostream>
2. <cctype>
3. <stdio.h>
4. <stdlib.h>
5. <conio.h>
6. <fstream>
7. <string.h>
8. + η βιβλιοθήκη "ClassInvertedIndex.h" προκειμένου να επικοινωνεί το πρόγραμμα με την κλάση "ClassInvertedIndex" που δημιουργήσαμε.

Οι έτοιμες συναρτήσεις που χρησιμοποιήθηκαν στο πρόγραμμα είναι οι εξής :

1. ifstream
2. ofstream
3. strcmp
4. fopen
5. fclose
6. getline
7. fcsanf
8. feof
9. cerr
10. cout

Οι τύποι δεδομένων που χρησιμοποιήθηκαν στο πρόγραμμα είναι οι εξής :

1. int
2. char
3. string

Επίσης χρησιμοποιήθηκαν οι δομές επανάληψης for και while καθώς και η δομή επιλογής if..else .

Για την υλοποίηση του προγράμματος έχουμε δημιουργήσει μία κλάση η οποία περιλαμβάνει 6 συναρτήσεις, 4 structs και την συνάρτηση της main.

Στην κλάση "ClassInvertedIndex" περιλαμβάνονται οι εξής συναρτήσεις :

1. void ClassInvertedIndex::Eisagogi(int x,nodeptr &p), με την οποία πραγματοποιείται εισαγωγή νέου κόμβου στο δέντρο AVL.
2. int ClassInvertedIndex::YpsosAVL(nodeptr p), με την οποία υπολογίζεται το ύψος του AVL δέντρου.
3. void ClassInvertedIndex::DiagrafiDiagrafi(int, nodeptr &,int &x), με την οποία πραγματοποιείται διαγραφή ενός κόμβου από το δέντρο AVL.
4. void ClassInvertedIndex:: (nodeptr p,std::ofstream & ofs,int &x), με την οποία τυπώνονται στο αρχείο εξόδου "output.txt." σε αύξουσα σειρά οι γειτονικοί κόμβοι ενός AVL δεντρου.
5. int ClassInvertedIndex::max(int value1, int value2), με την οποία υπολογίζουμε ποια από τις 2 μεταβλητές έχει την μεγαλύτερη τιμή.
6. int ClassInvertedIndex::DiagrafiElaxistou(nodeptr &p), με την οποία διαγράφουμε τον κόμβο, του AVL δέντρου, με την ελάχιστη τιμη.

Στην κλάση "ClassInvertedIndex" περιλαμβάνονται τα εξής structs :

1. nodeptr AristeriPeristrofi(nodeptr &),με το οποίο πραγματοποιείται αριστερή περιστοφή όταν υπερφορτώνεται η δεξία πλευρά του δεξιού υποδέντρου.
2. nodeptr DipliAristera(nodeptr &), με το οποίο πραγματοποιείται πρώτα μία δεξιά περιστροφή και στην συνέχεια μία αριστερή περιστροφή, όταν υπερφορτώνεται η αριστερή πλευρά του δεξιού υποδέντρου.
3. nodeptr DeksiaPeristrofi(nodeptr &),&),με το οποίο πραγματοποιείται δεξιά περιστροφή όταν υπερφορτώνεται η αριστερή πλευρά του αριστερού υποδέντρου.
4. nodeptr DipliDeksia(nodeptr &), με το οποίο πραγματοποιείται πρώτα μία αριστερή περιστροφή και στην συνέχεια μία δεξιά περιστροφή, όταν υπερφορτώνεται η δεξιά πλευρά του αριστερού υποδέντρου.

Αρχικά η συνάρτηση main ανοίγει το αρχείο "commands.txt" μόνο για ανάγνωση. Έπειτα διαβάζει διαδοχικά της σειρές του αρχείου. Αν διαβαστεί η συμβολοσειρά "READ\_DATA input.txt" τότε η συνάρτηση main καλώντας τις κατάλληλες συναρτήσεις ανοίγει το αρχείο "input.txt." μόνο για ανάγνωση και από αυτό εισάγει όλους τους κόμβους στο AVL δέντρο και στη συνέχεια δημιουργεί ένα ξεχωριστό AVL δέντρο για κάθε κόμβο του AVL στο όποιο εισάγει όλους τους γειτονικούς κόμβους του κόμβου. Αν διαβαστεί η συμβολοσειρά "INSERT\_LINK x y" τότε η συνάρτηση main καλώντας τις κατάλληλες συναρτήσεις εισάγει τον κόμβο y στο AVL δέντρο του κόμβου x. Αν διαβαστεί η συμβολοσειρά "DELETE\_LINK x y" τότε η συνάρτηση main καλώντας τις κατάλληλες συναρτήσεις διαγράφει τον κόμβο y από το AVL δέντρο του κόμβου x. Αν διαβαστεί η συμβολοσειρά

"WRITE\_INDEX output.txt" τότε η συνάρτηση main καλώντας τις κατάλληλες συναρτήσεις τυπώνει στο αρχείο "output.txt" πρώτα τον αριθμό του κόμβου, έπειτα το πλήθος των γειτόνων του και στην συνέχεια τυπώνει με αύξουσα σειρά τους γείτονες του κόμβου.