Τμήμα Μηχανικών Η/Υ & Πληροφορικής Μάθημα: Δομές Δεδομένων Εργασία

2016-2017

Διαμαντόπουλος Άγγελος ΑΜ. 5752

Email : diamant@ceid.upatras.gr

Γλώσσα προγραμματισμού : C++

1) Εισαγωγή

Σε αυτό το project , μας ζητείται να γραφτεί ένα λογισμικό για την οργάνωση στοιχείων ξενοδοχείων. Το project χωρίζεται σε μέρη ανάλογα με τα ερωτήματα του project , που θα αναλυθούν παρακάτω. Ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η c++ λόγω των δυνατοτήτων της , της ταχύτητας της, καθώς και της επιθυμίας μου για εξάσκηση σε αυτήν.

Θέλω να επισημάνω ότι υπάρχουν πολλά σχόλια στον κώδικα (σχεδόν ανά γραμμή) που θα κάνουν πιο κατανοητό τον τρόπο σκέψης μου στην επίλυση των ερωτημάτων για αυτό το λόγο δεν αναφέρω αναλυτικά κάποιες διαδικασίες στην αναφορά, καθώς μάλλον περισσότερο μπέρδεμα θα φέρουν, παρά ξεκαθάρισμα, λόγω της έλειψης κώδικα. Τέλος, επισημαίνω πως μετά το ερώτημα Ε (αυτό με τις μετρήσεις) , και πριν την εισαγωγή του κώδικα στην αναφορά, θα έχω κάποιες παρατηρήσεις και λεπτομέρειες για το πρόγραμμα.

2) ΜΕΡΟΣ Α

Το πρώτο μέρος της εργασίας ,αποτελεί και τη βάση της. Αρχικά , καλούμαστε να δημιουργήσουμε 2 structs , Hotel και Reservation , όπου το καθένα θα έχει τις δικές του ιδιότητες. Στη συνέχεια , καλούμαστε να φτιάξουμε το μενού με τις αντίστοιχες λειτουργείες του. Οι διαδικασίες απαριθμούνται και αναφέρονται αναλυτικά παρακάτω.

1. *Load Hotels and Reservations from file*

Πρέπει να περάσουμε τα στοιχεία του αρχείου, στην μνήμη του υπολογιστή . Για να γίνει αυτό, χρησιμοποιώ έναν τρόπο που χρησιμοποιώ αντίστοιχα και σε υπόλοιπα ερωτήματα, για αυτό το λόγο θα την αναπτύξω μόνο εδώ. Αυτός ο τρόπος είναι η χρήση vectors. Προτιμάται η χρήση vectors , από αυτή των arrays , για δύο λόγους:

- Η επεξεργασία τους είναι δυναμική , συνεπώς δεν χρειάζεται να γίνει malloc διαρκώς, πράγμα που θα ήταν απαραίτητο με τη χρήση πινάκων.
- Δεν υπάρχει κάποιο όριο στη μνήμη που πρέπει να δεσμευτεί για αυτούς.

Συγκεκριμένα , κάνω το εξής :

Προσπελάζω το αρχείο ως προς γραμμές , και ως προς κελιά. Δηλαδή μέσω iterators , αρχικά μπαίνω σε μια γραμμή, την προσπελάζω ως προς τα κελιά, και κάθε στοιχείο κελιού, το εισάγω σε ένα vector. Έπειτα , πριν συνεχίσω στην επόμενη γραμμή, εισάγω τον vector που μόλις έβαλα τα στοιχεία των κελιών σε αυτόν, σε έναν άλλον vector (vector μέσα σε vector). Ακολουθώ την ίδια διαδικασία μέχρι να προσπελαστεί όλο το αρχείο, και έτσι φορτώνω όλα τα στοιχεία του αρχείου στην μνήμη του υπολογιστή.

2. *Save Hotels and Reservations to file*

Αρχικά, δημιουργώ το αρχείο στο οποίο θέλω να γίνει η αποθήκευση (για τον τρόπο βλ. κώδικα, υπάρχουν σχόλια).

Έπειτα, ακολουθώντας διαδικασία με αυτή που εξήγησα προηγουμένως, έχω όλα τα στοιχεία σε vectors στην μνήμη του υπολογιστή . Ανοίγω το αρχείο που θέλω να περάσω τα στοιχεία σε αυτό (default όνομα είναι το : new.csv) με τρόπο που φαίνεται στον κώδικα , και κάνοντας iteration στους vectors, γράφω τα στοιχεία τους στο αρχείο, ενώ όπου πρέπει βάζω κόμματα και end lines .

3. *Add a Hotel (μαζί και τις κρατήσεις του)*

Με αυτή την επιλογή, πρέπει να εισάγουμε στο αρχείο μας, νέα ξενοδοχεία, καθώς και κρατήσεις σε αυτά. Για να το κάνουμε αυτό, αρχικά ανοίγουμε το αρχείο μας, και ενημερώνουμε την τιμή της πρώτης γραμμής (που αναφέρει

τον αριθμό των ξενοδοχείων στο αρχείο) προσθέτοντας +1 στην τιμή που έχει εκείνη τη στιγμή. Αυτό σημαίνει ότι θα προστεθεί ένα ξενοδοχείο με πολλές κρατήσεις. Κάθε φορά που θα καλούμε αυτή τη συνάρτηση λοιπόν, θα αυξάνεται αυτή η τιμή κατά 1. Αφού γίνει η ενημέρωση, ο χρήστης, καλείται να δώσει τα στοιχεία του ξενοδοχείου, καθώς και της κράτησης σε αυτό. Για να γίνει αυτό χρησιμοποιούμε δείκτες τύπου hotel και reservation (βλ. κώδικα).

4. *Search and Display a Hotel by id*

Εδώ ζητείται η αναζήτηση ενός ξενοδοχείου από το id του. Σε αυτή την επιλογή έχω προσθέσει άλλες 4 υποεπιλογές , για τον τρόπο αναζήτησης (linear, binary , interpolation , AVL) . Σε αυτή τη φάση θα αναλύσω τον γραμμικό τρόπο αναζήτησης . Οι υπόλοιποι τρόποι θα αναφερθούν παρακάτω. Αρχικά χρησιμοποιώ τη διαδικασία που ανέφερα παραπάνω και περνάω τα στοιχεία από το αρχείο στην μνήμη του υπολογιστή. Έπειτα, δίνω το id ως όρισμα τύπου int στην συνάρτηση που είναι υπεύθηνη για την γραμμική αναζήτηση . Όμως, επειδή τα στοιχεια περνάνε στους vector ως strings , μετατρέπω την μετβλητή -όρισμα από int σε string και έπειτα κάνω την αναζήτηση . Η αναζήτηση γίνεται με iterate στους vectors, κοιτάζοντας μόνο τις θέσεις των id τους . Όταν κάποιο id είναι ίδιο με αυτό που δώσαμε ως όρισμα ,τότε σταματάμε την προσπέλαση , και εκτυπώνουμε στον χρήστη πως το id που θέλει βρέθηκε. Αν πάλι προσπελαστούν οι vectors σε ολόκληρο το μέγεθός τους και δεν βρεθεί το id που θέλουμε, ενημερώνουμε τον χρήστη πως δεν βρέθηκε.

5. *Display Reservations by surname search*

Σε αυτό το ερώτημα , ζητείται να κάνουμε αναζήτηση κρατήσεων με βάση το επώνυμο στο οποίο είναι κλεισμένη η κράτηση. Και πάλι έχω 2 υποεπιλογές, μια για γραμμική αναζήτηση, και μια για αναζήτηση μέσω Trie. Εδώ θα αναλύσω την γραμμική. Γενικά ακολουθούμε παρόμοια διαδικασία με αυτή για την αναζήτηση μέσω id, απλά στην προκειμένη περίπτωση ελέγχουμε την θέση στην οποία βρίσκονται τα ονόματα στο αρχείο μας (ή στους vector μετά την φόρτωση του αρχείου σε αυτούς).

3) ΜΕΡΟΣ Β

Σε αυτό το ερώτημα , καλούμαστε να υλοποιήσουμε αναζήτηση με βάση το id με την μέθοδο του binary search , και του interpolation search.

Binary Search:

Για να χρησιμοποιήσουμε αυτή τη μέθοδο, αρχικά , περνάμε το αρχείο μας στην μνήμη του υπολογιστή. Έπειτα, κάνουμε ταξινόμηση των στοιχείων με βάση το id. Για την ταξινόμηση, προτίμησα να χρησιμοποιήσω την συνάρτηση sort της stl , καθώς είναι πολύ εύχρηστη και γρήγορη. Στην συνάρτηση sorting, προκειμένου να γίνει σωστά η σύγκριση των id , σύγκρινα τα string id's αφού πρώτα τα μετέτρεψα σε αριθμούς (int) , μέσω της stoi() συνάρτησης (βλ. κώδικα). Αφού έγινε η σύγκριση , ορίζω 3 int μεταβλητές, τις : start , end , mid και 3 string μεταβλητές , τις : start_s , end_s , mid_s . Οι int μεταβλητές αντιπροσωπεύουν τα indexes των στοιχείων που θα ψάξουμε, ενώ τα strings , τις τιμές που έχουν τα συγκεκριμένα indexes στο πεδίο των id. Έπειτα, αναδρομικά , γίνεται η κλασική διαδικασία του binary search . Δηλαδή , χρησιμοποιούμε τις παραπάνω μεταβλητές για τον προσδιορισμό του επόμενου στοιχείου για αναζήτηση . Αν το στοιχείο που προκύψει είναι μεγαλύτερο από το mid στοιχείο, τότε το low = mid και ξαναυπολογίζουμε το mid με βάση τα νέα δεδομένα. Αντίστοιχα αν το στοιχείο είναι μικρότερο από το mid , απλά τότε κάνουμε το high = mid . Αν το στοιχείο που ψάχνουμε είναι ίδιο με το mid , τότε βρήκαμε αυτό που θέλαμε , ενώ αν η νέα τιμή του mid είναι ίδια 2 συνεχόμενες φορές, τότε ξέρουμε πως δεν υπάρχει το στοιχείο .Έχω σχόλια στον κώδικα για τον τρόπο που γίνεται η κάθε αναζήτηση, για αυτό το λόγο δεν αναφέρω πολλές λεπτομέρειες εδώ.

Interpolation Search:

Και σε αυτό τον τρόπο αναζήτησης , αρχικά περνάμε τα αρχεία στην μνήμη , και τα ταξινομούμε , ακολουθώντας την ίδια διαδικασία που έχω ακολουθήσει και στο binary search . Έπειτα, ορίζω 4 int μεταβλητές . Την start, που αντιπροσωπεύει το πρώτο index (αρχικά δηλαδή 0) , την end, που μας δείχνει το index του τελευταίου στοιχείου (αρχική τιμή , ανάλογα με το μέγεθος του vector που περιέχει τα στοιχεία μας) , την μεταβλητή e , η οποία περιέχει αρχικά την τιμή του id μας σε int μορφή , και τέλος η μεταβλητή mid , η οποία περιέχει την τιμή που υπολογίζουμε από τον τύπο για την interpolation αναζήτηση, δηλαδή τον εξής:

```
mid = start + (((end - start)*(e - stoi(vec_v[start][0]))) / (stoi(vec_v[end][0])
- stoi(vec_v[start][0])));
```

Και αναδρομικά στη συνέχεια γίνεται η γνωστή διαδικασία του interpolation search . Δηλαδή ,αν το mid = id που ψάχνουμε, βρήκαμε το στοιχείο που θέλαμε και σταματάμε την αναζήτηση . Αν είναι το mid μικρότερο από το id , υπολογίζουμε την νέα τιμή του mid , και ενημερώνουμε την νέα τιμή του start (για τη αντίθετη περίπτωση, κάνουμε το ίδιο , υπολογίζοντας την τιμή του end) .

4) ΜΕΡΟΣ Γ

Σε αυτό το ερώτημα καλούμαστε να φτιάξουμε ένα AVL δέντρο, και συγκεκριμένα τις διαδικασίες εισαγωγής και εύρεσης. Αρχικά , φτιάχνουμε ένα struct που περιέχει ιδιότητες που θα μας φανούν χρήσιμες στην διαδικασία κατασκευής του δέντρου (λεπτομέρειες στον κώδικα). Έπειτα, φτιάχνουμε μια συνάρτηση, η οποία θα μας επιστρέφει το ύψος του δέντρου. Επιπλέον δημιουργούμε την συνάρτηση newNode , η οποία θα επιστρέφει ένα Node του δέντρου, το οποίο θα το αρχικοποιεί με την τιμή του id που θέλουμε, ενώ θα αρχικοποιεί και τους left και right (βλ. struct Node) pointers του , με τιμή NULL. Τέλος αυτή η συνάρτηση ενημερώνει το ύψος του κόμβου. Έχουμε φτιάξει και μια συνάρτηση η οποία επιστρέφει το μέγιστο μεταξύ δύο ακέραιων αριθμών , η οποία θα μας φανεί χρήσιμη για τις επόμενες 2 συναρτήσεις που φτιάξαμε, την rightRotate, και την leftRotate , οι οποίες είναι υπεύθυνες για τα rotates που γίνονται κατά τη διάρκεια των εισαγωγών στο δέντρο μας. Η βασικότερη συνάρτηση, είναι η insert , η οποία βάζει ένα κόμβο στο δέντρο μας, τηρώντας τους κανόνες του AVL δέντρου. Για να το κάνει αυτό χρησιμοποιεί τις παραπάνω συναρτήσεις , καθώς και την getBalance (η οποία μας ενημερώνει αν έχει γίνει balanced το δέντρο μας μετά από κάποιο rotation). Όλες αυτές οι συναρτήσεις είναι χρήσιμες για την διαδικασία εισαγωγής κόμβου στο δέντρο. Για την αναζήτηση σε αυτό χρησιμοποιούμε την search_avl , μια αναδρομική συνάρτηση , η οποία επιστρέφει 1 αν βρέθηκε το στοιχείο μας, ή 0 αν δεν βρέθηκε. Τέλος η συνάρτηση preOrder , μας βοηθάει στην αναπαράσταση του δέντρου(για testing κυρίως) ενώ η συνάρτηση AVL , είναι αυτή που συνδέει όλες τις παραπάνω λειτουργείες (επαναλαμβάνω πως δεν αναφέρω αναλυτικές λεπτομέρειες εδώ για την ακριβές λειτουργεία του κάθε αλγόριθμου, καθώς υπάρχουν λεπτομέριες στον κώδικα).

5) ΜΕΡΟΣ Δ

Σε αυτό το ερώτημα καλούμαστε να φτιάξουμε ένα ψηφιακό δέντρο (trie) . Συγκεκριμένα , πρέπει να κάνουμε αναζητήσεις με βάση τα ονόματα τω κρατήσεων , καθώς και να φορτώνουμε όλες τις κρατήσεις από όλα τα ξενοδοχεία σε αυτό. Αρχικά φτιάχνουμε ένα struct που ονομάζουμε node και περιέχει διάφορες χρήσιμες ιδιότητες που θα μας χρειαστούν. Συγκεκριμένα έχει έναν vector v , ο οποίος αν είναι φύλο ο συγκεκριμένος κόμβος, θα τον αποθηκεύσει, μαζί με τα στοιχεία του ,ως προς το όνομα του ξενοδοχείο με βάση το όνομα του πελάτη, και έναν vector k , ο οποίος περιέχει στοιχεία του πελάτη στον οποίο έχει γίνει η κράτηση , με σκοπό την εκτύπωση των στοιχείων αυτών. Επιπλέον έχουμε τον πίνακα δεικτών child ,ο οποίος δείχνει σε κάθε γράμμα που έχουμε στο αλφάβητο που διαχειριζόμαστε . Η int μεταβλητή *prefix_count* μετράει τον αριθμό των λέξεων στο λεξικό μας , που έχουν ένα συγκεκριμένο string ως πρόθεμα ήδη. Έπειτα έχουμε τη συνάρτηση init() η οποία αρχικοπειεί το Trie δημιουργώντας ένα νέο στοιχείο τύπου node και καταλαμβάνοντας στην μνήμη όσο χώρο χρειάζεται, αρχικοποιώντας για αυτό την τιμή prefix_count = 0 αφού μόλις δημιουργήθηκε, άρα δεν έχει προθέματα και τέλος ενημερώνοντας πως δεν είναι leaf , θέτοντας τον υπεύθυνο για αυτή την πληροφορία δείκτη ίσο με NULL . Στη συνέχεια ,ορίζουμε την βασική συνάρτηση, την insert. Μέσω της insert , εισάγω στο δέντρο τις λέξεις που πρέπει ,δηλαδή τα ονόματα των ξενοδοχείων τα στοιχεία των κρατήσεων και για ένα γράμμα τη φορά το μετατρέπει σε αριθμό για τσεκάρει αν ο δείκτης που αντιστοιχεί σε αυτό είναι μαρκαρισμένος ή όχι. Αν δεν είναι μαρκαρισμένος , εισάγεται αυτός ο κόμβος στο δέντρο γράμμα-γράμμα , και στο τελευταίο γράμμα αποθηκεύει την λέξει ως leaf (true ο υπεύθυνος δείκτης) . Αν υπάρχει ήδη κάποιο μέρος της λέξης αποθηκευμένο , τότε απλά γίνεται η παραπάνω διαδικασία για το πρώτο καινούργιο γράμμα, μέχρι το τέλος. Επίσης, στο όταν τελειώσω με τη δέσμευση των λέξεων , κάνω ελευθέρωση της μνήμης από αυτές μέσω μιας διαδικασίας που αναλύω στον κώδικα. Η συνάρτηση search, αναζητάει τη λέξη που του δίνουμε ως όρισμα , πάνω στο δέντρο που έχουμε φτιάξει, με τον γνωστό τρόπο λειτουργείας αναζήτησης των tries. Αν την βρει την ζητούμενη λέξη, τότε επιστρέφει ως αποτελέσμα τα ονόματα-όνομα των ξενοδοχείων (πολλών ή και ενός) που περιέχουν κράτηση με το όνομα που βρέθηκε. Τέλος , η συνάρτηση trie, απλά συγκροτεί και συγχρονίζει όλες τις παραπάνω διαδικασίες .

6) ΜΕΡΟΣ Ε

Σε αυτό το σημείο πρέπει να κάνουμε κάποια πειράματα με βάση το συγκεκριμένο αρχείο που μας δίνεται, υπολογίζοντας τις αποδόσεις του κάθε αλγορίθμου πάνω σε αυτό για τυχαία δεδομένα. Αρχικά κάνουμε πειράματα για τυχαίο id (int αριθμός) ξενοδοχείων συγκρίνοντας τους αλγορίθμους Linear Search , Binary Search , interpolation , AVL search , και έπειτα για τυχαίο υπάρχων string , που αναφέρεται σε όνομα κράτησης , με βάση τους αλγορίθμους Linear Search και digital Trie. Για τα πειράματα εκτελώ 1000 επαναλήψεις (1000 τυχαία id's και strings) και υπολογίζω τον μέσο όρο χρόνου (σε milliseconds) και συγκρίσεων που προκύπτει από κάθε αλγόριθμο. Κάνω αυτή τη διαδικασία 10 φορές και συλλέγω τα στοιχεία που φαίνονται στον παρακάτω πίνακα. Έπειτα από τους 10 αυτούς μέσους όρους , βρίσκω τον μέσο όρο τους, και φτιάχνω γραφικές παραστάσεις.

Ο τρόπος που βρίσκω τους μέσους όρους κατά την εκτέλεση του κώδικα, γίνεται με τη βοήθεια της συνάρτησης *metriseis* , με την εξής λογική .Αρχικά έχω κάνει define μια μεταβλητή η οποία ορίζει τον αριθμό επαναλήψεων των πειραμάτων (1000 στην περίπτωσή μας) . Έπειτα σε μια for , η οποία εκτελείται όσες φορές μας λέει η μεταβλητή που ανέφερα πριν , παράγω ένα τυχαίο int και ένα τυχαίο (υπαρκτό στο αρχείο , string) που αφορούν τα id των ξενοδοχείων και τα ονόματα των κρατήσεων αντίστοιχα , και τα δίνω ως εισόδους στις συναρτήσεις που αφορούν τους αλγορίθμους που έχουμε φτιάξει (για μεγαλύτερη ανάλυση βλ. κώδικα). Για κάθε αλγόριθμο που εκτελείται , αθροίζω τον αριθμό συγκρίσεων και τον χρόνο εκτέλεσης (στον χρόνο εκτέλεσης αφαιρώ τον χρόνο αρχικοποίησης κάθε αλγορίθμου, ώστε να συγκρίνω καθαρά και μόνο τον χρόνο εκτέλεσης) σε μεταβλητές . Όταν τελειώσει η for , έχω σε μεταβλητές τον συνολικό αριθμό συγκρίσεων/χρόνου εκτέλεσης του κάθε αλγορίθμου, και διαιρώντας με τον συνολικό αριθμό επαναλήψεων παράγω τον μέσο όρο. Τέλος, τα αποτελέσματα εκτυπώνονται σε πίνακα με την εξής μορφή :

Μαζεύω λοιπόν τα στοιχεία από 10 τέτοιου είδους μετρήσεις και τα παραθέτω παρακάτω.

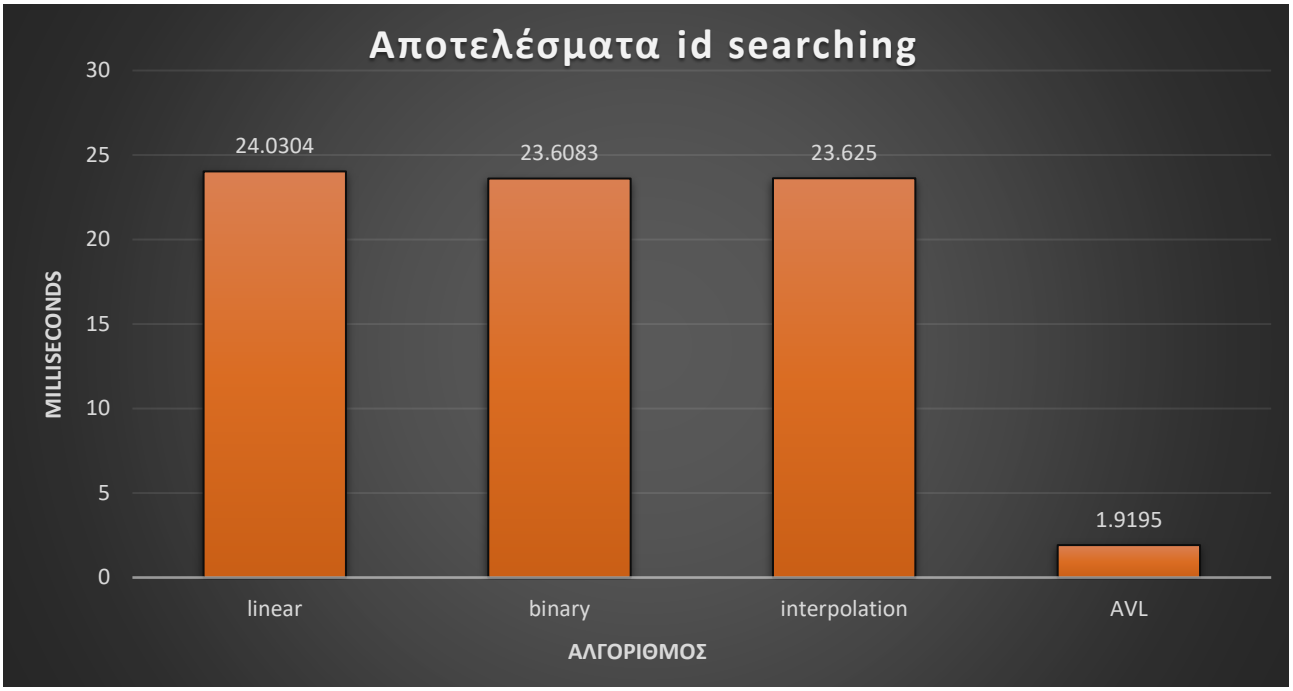
Όπως θα δούμε παρακάτω στις μετρήσεις , τα αποτελέσματα, βγαίνουν ακριβώς όπως τα περιμένουμε με βάση την θεωρεία. Σε κάθε περίπτωση , ο γραμμικός χρόνος αναζήτησης μας δίνει την χειρότερη απόδοση και ως προς τις συγκρίσεις αλλά και ως προς τον χρόνο σε σχέση με τους υπόλοιπους. Ο binary search αλγόριθμος, είναι καλύτερος από τον γραμμικό , και κατά ελάχιστα milliseconds καλύτερος από τον interpolation , πράγμα λογικό από τη στιγμή που έχουμε μικρό σχετικά αριθμό δεδομένων . Ο AVL όπως βλέπουμε έχει με διαφορά την καλύτερη απόδοση από θέμα χρόνου (αν και έχει μεγαλύτερο initialization time από τους άλλους). Όσο αφορά τώρα, την αναζήτηση των ονομάτων κρατήσεων, βλέπουμε πως ο Trie είναι κατά τεράστιο ποσοστό πολύ καλύτερος από τον linear τόσο στο χρόνο όσο και στον αριθμό συγκρίσεων, γεγονός αναμενόμενο.

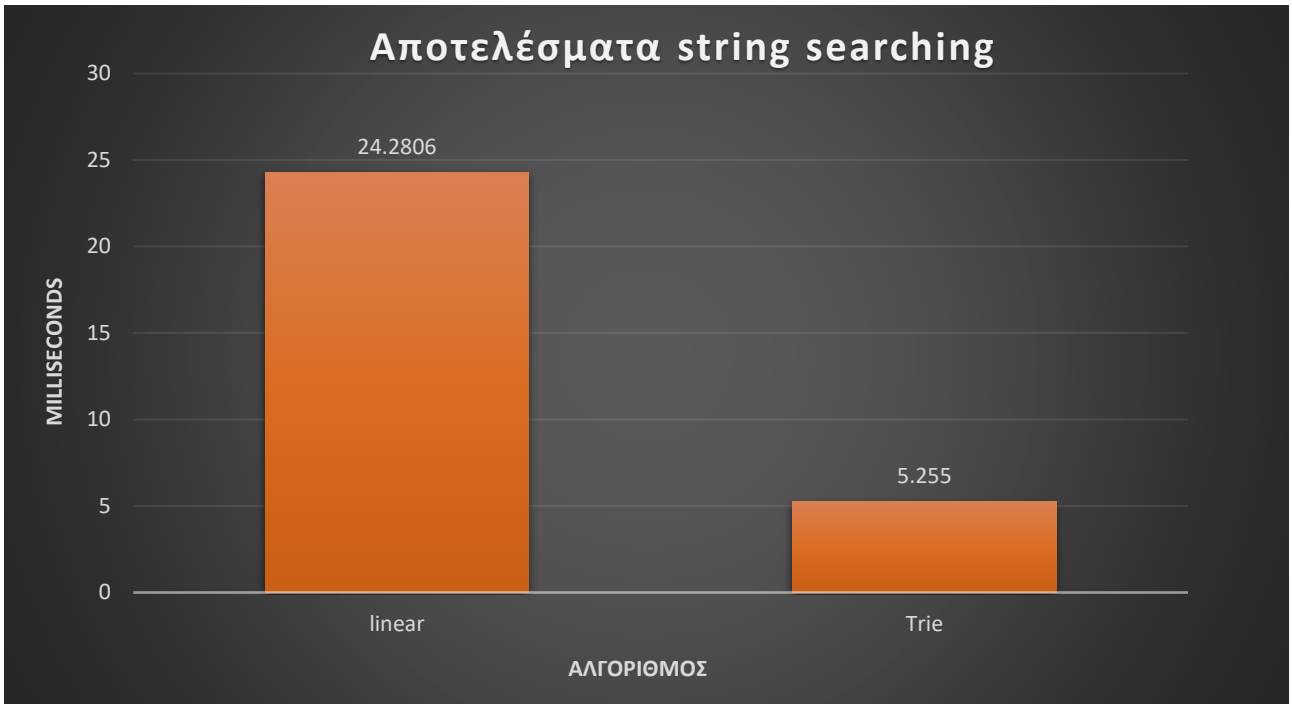
ΠΑΡΑΤΗΡΗΣΕΙΣ :

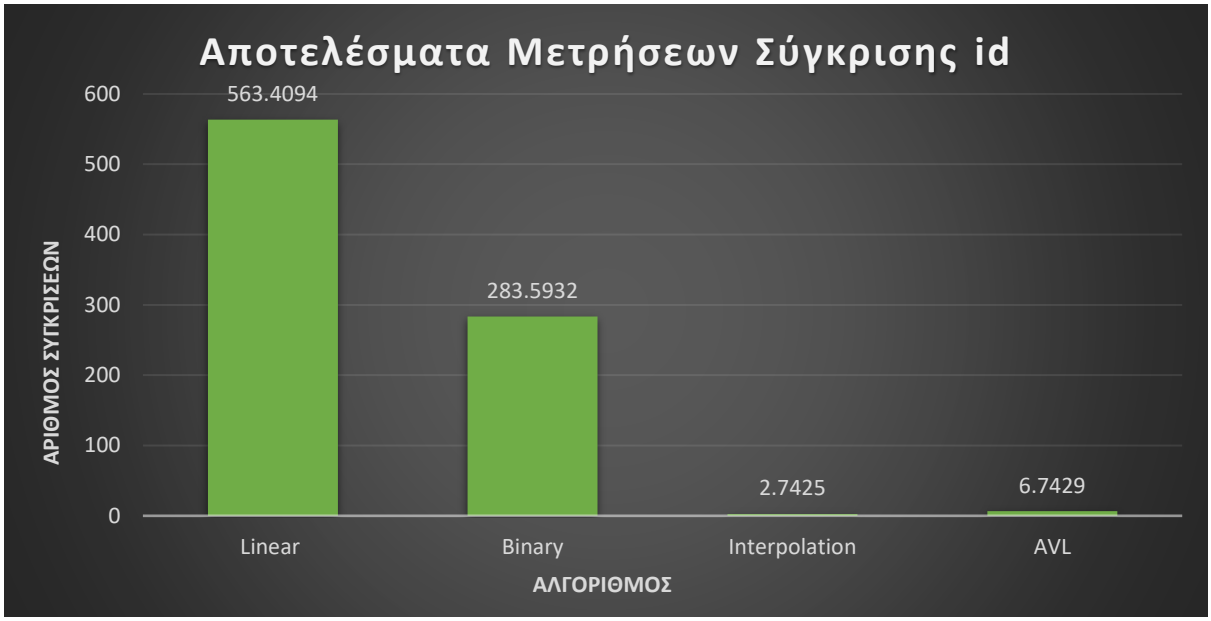
- Οι αλγόριθμοι τρέξανε σε Visual Studio 2017 στο release mode (το οποίο σε σχέση με το default mode που είναι το debug , είναι πάρα πολύ πιο γρήγορο. Για αυτό το λόγο τα milliseconds είναι μικρά)
- Στον κώδικα δεν υπάρχουν καθόλου memory leaks.
- Στο μενού επιλογών , θεώρησα σωστό να βάλω μια έξτρα επιλογή , την επιλογή *metriseis* (επιλογή 6) , η οποία εκτελεί τα πειράματα και εμφανίζει τα αποτελέσματα στην μορφή του πίνακα που έδειξα παραπάνω. Επέλεξα να το αφήσω στο μενού , ώστε να μπορεί όποιος θέλει να επιβεβαιώσει ότι τα αποτελέσματα των πειραμάτων μου είναι αληθινά και δεν τα έχω επιλέξει εγώ αυθαίρετα.
- Ο κώδικας έχε πάρα πολλά σχόλια (σχεδόν γραμμή παρά γραμμή) τα οποία καθιστούν εύκολη την κατανόηση της λογικής που ακολούθησα για κάθε ερώτημα. Για αυτό δεν έχω αναλύσει με πολύ μεγάλη λεπτομέρεια στην αναφορά κάθε τι που κάνω , καθώς δεν θα είχε νόημα.

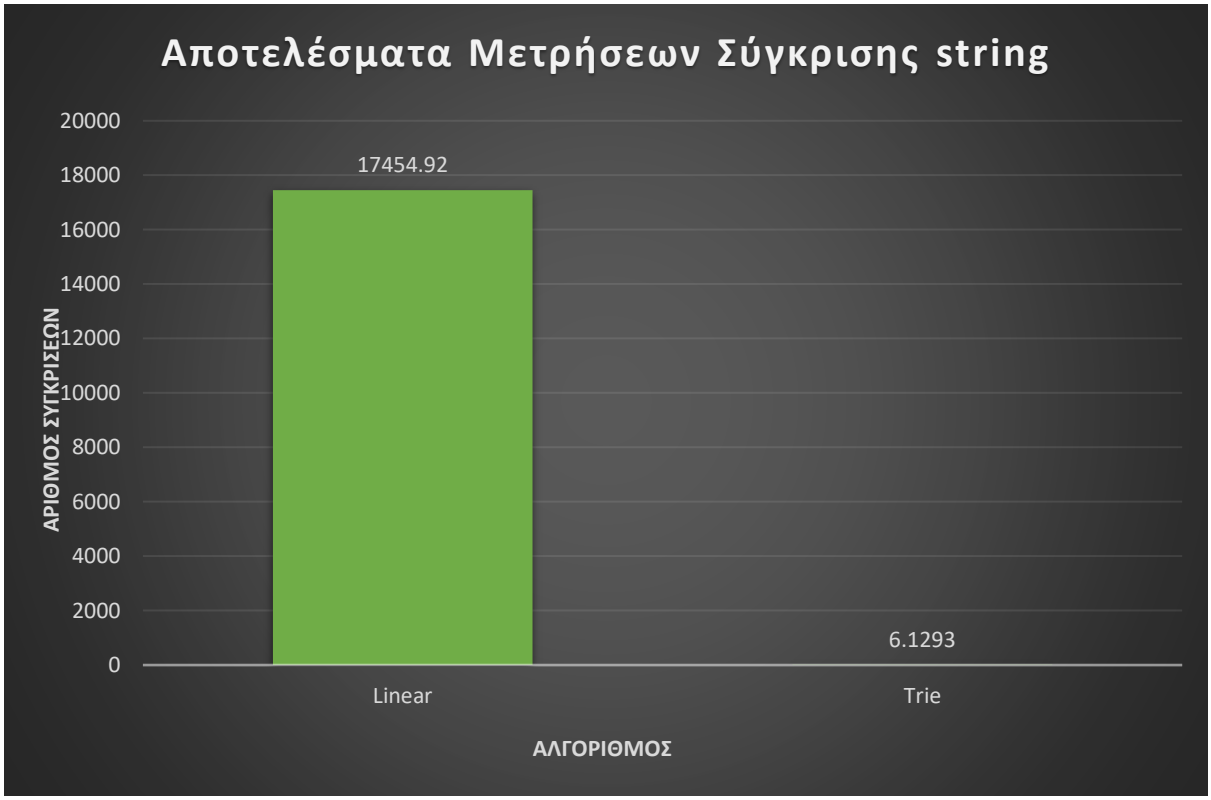
Αριθμός Πειραμάτων	1η	2η	3η	4η	5η	6η	7η	8η	9η	10η	Average	Μέθοδος
Χρόνος σε millisecond	23.704	24.184	23.953	23.968	24.004	24.349	24.08	24.009	24.029	24.024	24.0304	Linear Search (για id)
Χρόνος σε millisecond	23.509	23.986	23.724	23.622	23.679	23.496	23.634	23.588	23.421	23.424	23.6083	Binary Search (για id)
Χρόνος σε millisecond	23.45	24.084	23.82	23.826	23.628	23.364	23.505	23.56	23.385	23.628	23.625	Interpolation Search (για id)
Χρόνος σε millisecond	1.989	1.893	1.936	1.973	1.883	2.023	1.884	1.84	1.888	1.886	1.9195	AVL Tree (για id)
Χρόνος σε millisecond	25.919	25.148	24.063	23.913	23.075	23.694	24.527	23.887	24.192	24.388	24.2806	Linear Search (για string)
Χρόνος σε millisecond	5.802	5.492	5.298	5.257	5.07	5.129	5.103	5.19	5.083	5.126	5.255	Digital Trie (για string)

Αριθμός Πειραμάτων	1η	2η	3η	4η	5η	6η	7η	8η	9η	10η	Average	Μέθοδος
Αριθμός Συγκρίσεων	567.959	559.526	568.799	567.434	577.899	564.223	544.699	542.691	579.014	561.85	563.4094	Linear Search (για id)
Αριθμός Συγκρίσεων	254.814	236.508	242.342	247.28	626.728	251.804	235.23	213.008	270.076	258.142	283.5932	Binary Search (για id)
Αριθμός Συγκρίσεων	2.739	2.714	2.733	2.774	2.748	2.721	2.737	2.76	2.773	2.726	2.7425	Interpolation Search (για id)
Αριθμός Συγκρίσεων	6.74	6.911	6.801	6.751	6.642	6.724	6.641	6.714	6.518	6.987	6.7429	AVL Tree (για id)
Αριθμός Συγκρίσεων	17455.3	17452.7	17457.9	17453.9	17455.7	17454.2	17455	17454.8	17455.2	17454.5	17454.92	Linear Search (για string)
Αριθμός Συγκρίσεων	6.121	6.132	6.067	6.08	6.186	6.153	6.138	6.086	6.232	6.098	6.1293	Digital Trie (για string)


Αποτελέσματα id searching
30
25
20
15
10
5
0
MILLISECONDS
24.0304
23.6083
23.625
1.9195
linear
binary
interpolation
AVL
ΑΛΓΟΡΙΘΜΟΣ


Αποτελέσματα string searching
30
25
20
15
10
5
0
MILLISECONDS
24.2806
5.255
linear
Trie
ΑΛΓΟΡΙΘΜΟΣ


Αποτελέσματα Μετρήσεων Σύγκρισης id
600
500
400
300
200
100
0
563.4094
283.5932
2.7425
6.7429
ΑΡΙΘΜΟΣ ΣΥΓΚΡΙΣΕΩΝ
Linear
Binary
Interpolation
AVL
ΑΛΓΟΡΙΘΜΟΣ


Αποτελέσματα Μετρήσεων Σύγκρισης string
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
17454.92
6.1293
ΑΡΙΘΜΟΣ ΣΥΓΚΡΙΣΕΩΝ
Linear
Trie
ΑΛΓΟΡΙΘΜΟΣ

ΚΩΔΙΚΑΣ ΑΣΚΗΣΗΣ

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include <stdio.h>
#include <stdlib.h>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/chrono.hpp>
#include <algorithm>
#include <ctime>


#define EPANAL 1000   // arithmos epanalipsewn gia testings

using namespace std;
using namespace boost;

// parakatw einai metavlites pou tha xrisimopoiithoun gia ta apotlesmata twn peiramatwn .

double binary_sorting;  // xronos gia sorting timwn , stin binary anazitisi
double inter_sorting;   // xronos gia sorting timwn , stin interpolation anazitisi
double AVL_init;       //xronos gia arxikopoiisi tou AVL dentrou
double Trie_init;       //xronos gia arxikopoiisi tou Trie dentrou
double liniear_sur_init;
double Trie_init2;
```

```
double counterBinary = 0;   // arithmos sigkrisewn mexri na vrethei to apotelesma stin binary anazitisi
double counterSearch = 0;    // arithmos sigkrisewn mexri na vrethei to apotelesma stin grammiki anazitisi
double interpolSearch = 0;    // arithmos sigkrisewn mexri na vrethei to apotelesma stin interpolation anazitisi
double AVLsearch = 0;         //  arithmos sigkrisewn mexri na vrethei to apotelesma stin anazitisi sto AVL dentro
double TrieSearch = 0;         // arithmos sigkrisewn mexri na vrethei to apotelesma stin anazitisi tou Trie
double liniear_sur_comp = 0;



vector<vector<string>> file_to_vector(ifstream& file, string& fname);    // auti i sinartisi , tha pernaei ola ta stoixeia tou arxeio se vector, ektos apo tin prwti grammi (ton arithmo twn eisagwgwn)
vector<vector<string>> file_to_vector2(ifstream& file, string& fname);  // auti tha pernaei kai tin prwti grammi


struct Reservation {      // morfi tou struct gia Reservation
        string name;          //onoma kratisis
        string checkinDate;     // imerominia check in
        int stayDurationDays;   // poses meres tha meinoun ta atoma sto dwmatio tou ksenodoxeiou
};


struct Hotel {  // morfi struct gia Hotel
        int id;             // to id tou ksendoxeiou
        string name;        // to onoma tou ksenodoxeiou
        int stars;          // arithmos asteriwn ou eexei to ksenodoxeio
        int numberOfRooms;   // arithmos dwmatiwn
        Reservation* reservations;   // deiktis pros kratisi
};


void print_vectors(vector<vector<string>>& vec_v, int size ) {  // ektipwsi opoiou vector tou dwsoume ws orisma. Xrisimo gia testings.
        int k = 0;
                for (vector<vector<string>>::iterator it = vec_v.begin(); it != vec_v.end(); ++it) {   // o iterator einai vector<strings>
                        for (int i = 0; i<(*it).size();++i ) {   // diapername ta stoixeia tou kathe iterator, diladi ta strings ta opoia apoteloun ta stoixeia tou arxeou
```

```
                    cout << "vector [" << k << "]" << "  " << (*it)[i] << endl;
                }
                k++;
            }
}

int get_new_id(ifstream & file , string& fname) {  //epistrefei monadiko id se kathe neo entry sto arxeio mas

    if (file.is_open() == 0 ) {   // an den einai anoixto idi to arxeio, tote to anoigoume wste na mporoume na to epeksergastoume
        file.open(fname);
    }

    string line;
    int temp = 0;
    int counter = 0;

    vector<vector<string>> ids = file_to_vector(file, fname) ;  //o vector ids periexei ta stoixeia tou arxeio

    for (vector<vector<string>>::iterator it = ids.begin(); it != ids.end(); it++ ) {   // vriskoume to id me to megalitero arithmo , kai to auksanoume kata 1.
        if ( stoi( (*it)[0] ) >= temp) {                    // etsi, epistrefoume tin timi pou proekipse, i opoia einai kai monadiki
            temp = stoi((*it)[0]);
        }
    }

    temp = temp + 1;   // auksanei kata ena tin current megisti timi

    if (file.is_open()==1) {   //kleinoume t arxeio an einai anoixto
        file.close();
    }

    return temp;   // epistrefoume tin nea monadiki timi pou tha einai to id
```

```
}

void add_hotel(ifstream& file , ofstream& out , string fname ){  // vazoume ksenodoxeio sto arxeio pou dinoume ws orisma, me stoixeia pou theloume emeis

        vector<vector<string>> vec_v;   // edw tha perasoume ola ta stoixeia tou arxeiou
        vec_v = file_to_vector2(file, fname);



        if (file.is_open() == 0) {     // anoigoume to arxeio an den einai idi anoixto
                file.open(fname);
        }
        ofstream temp;                  // theloume na allaksoume tin prwti grammi , wste na anagrafei ton arithmo twn kataxwrisewn meta tin prosthiki tou ksenodoxeiou.
        if (temp.is_open() == 0) {          // anoigoume to ostream gia na grapsoume sto arxeio
                temp.open(fname);
        }

        int counter = 0;          // me ti voitheia tou counter tha allaksoume tin timi tis prwtis grammis
        int si = vec_v.size();
        si = si + 1;
        for (vector<vector<string>>::iterator it = vec_v.begin(); it != vec_v.end(); it++) {    // iteration sto arxeio

                if (counter == 0) {             // allazoume tin prwti grammi
                        temp << si  << ";" << endl;
                        counter++;
                }
                else {

                        for (int i = 0; i < (*it).size(); i++) {   // pername kai ta ipoloipa stoixeia

                                if (i < (*it).size() - 1) {
                                        temp << (*it)[i] << ";";
                                }
```

```
                                else {
                                        temp << (*it)[i] << endl;

                                }
                        }

                }
        }

        if (file.is_open() == 1) {    // kleinoume to arxeio , me allagmeni tin prwi grammi
                file.close();
        }
        if (temp.is_open()==1) {    // kleinoume kai to ostream
                temp.close();
        }



        if (file.is_open() == 0 ) {          // ta anoigoume ksana gia na prosthesoume to neo ksenodoxeio
                file.open(fname);
        }

        if (out.is_open() == 0) {            // anoigoume kai to ostream
                out.open(fname);
        }


        Hotel* h = new Hotel;                 // deiktis se hotel tha mas voithisei gia na ekxwrisoume ta stoixeia
tou ksenodoxeiou
        Reservation *r = new Reservation;       // antistoixa gia ta asoixeia twn kratisewn
        h->reservations = r;
        h->id =  get_new_id(file, fname);    // mesw tis parapanw sinartisis pairnoume neo monadiko id
```

```
        out << h->id << ";";                    // zitame to xristi na mas dwsei ta stoixeia tou ksenodoxeiou
        cout << " Dwste mou to onoma tou ksenodoxeiou :" << endl;
        cin.ignore();
        getline(cin , h->name);
        out << h->name << ";";
        cout << "Dwste mou ton arithmo dwmatiwn tou ksenodoxeiou :" << endl;
        cin >> h->numberOfRooms;
        out << h->numberOfRooms << ";";
        cout << "Dwste mou ton arithmo twn asteriwn pou exei to ksenodoxeio: " << endl;
        cin >> h->stars;
        out << h->stars << ";";

        int kratisi = 1;
        while (kratisi == 1) {                  // afou mas dwsei ta stoixeia tou ksenodoxeiou, ton rwtame gia ta stoixea tis(twn) kratisis(kratisewn)
                cout << "Poio einai to onoma autou pou thelei na kanei tin kratisi ?" << endl;
                cin.ignore();
                getline( cin , h->reservations->name);
                out << h->reservations->name << ";";
                cout << "Poia imerominia thelete na to kleisete ?" << endl;
                cin >> h->reservations->checkinDate;
                out << h->reservations->checkinDate << ";";
                cout << "Poses meres tha katsete?" << endl;
                cin >> h->reservations->stayDurationDays;
                cout << " An thelete na kanete kialli kratisi pliktrologiste to 1, alliws to 0." << endl;
                cin >> kratisi;
                if (kratisi == 0) {
                        out << h->reservations->stayDurationDays << endl;
                }
                else {
                        out << h->reservations->stayDurationDays << ";" ;
                }
```

```
	}



	if (out.is_open() == 1) {   // kleinoume to arxeio ostream
		out.close();
	}
	if (file.is_open() == 1) {    // kleinoume kai to ifstream
		file.close();
	}

}

void load_hotels(ifstream& file , string& fname) {   //mpainoun ta ksenodoxeia apo to arxeio pou dinoume ws orisma , stin mnimi

	if (file.is_open() == 0) {
		file.open(fname);        // anoigoume arxeio
	}

	string line;      // diadikasia gia peramsa twn stoixeiown apo to arxeio sti mnimi
	int size;
	string temp;
	int moder = 0;
	stringstream lineStream(line);
	string cell;

	while (getline(file, line))
	{
		std::stringstream  lineStream(line);
		std::string       cell;
		while (std::getline(lineStream, cell, ';'))   // too kathe cell periexei string . Se auti tin periptwsi koitame tin prwti grammi mono
		{
```

```
                temp = cell;
                break;
            }
            break;
        }


    vector<vector<string>> vec_v;
    int l = 0;
    while (getline(file, line)) {                   // edw skanaroume to ipoloipo arxieo kai pername s vector ta stoixeia tou
            vector<string> v;
            std::stringstream  lineStream(line);
            std::string       cell;
            while (getline(lineStream, cell, ';'))    // arxika pername ta stoixeia twn grammwn ston vector v
            {

                v.push_back(cell);

            }
            vec_v.push_back(v);      // otan teleiwnoume me mia grammi stoixeiwn , pername auti ti grammi se vector
    }
    cout << "An thelete na ektipwste to arxeio sto console, patiste 1, alliws 0" << endl;
    int chooserr;
    cin >> chooserr;
    if (chooserr == 1) {
            print_vectors(vec_v, vec_v.size());
    }

    file.close();
    if ( file.is_open() == 0 ) { cout << "vector size : "<< vec_v.size() << endl; }
```

```
}

void search_display_only(vector<vector<string>>& vec_v, int id) {  // ektipwnei tis leptomeries tou ksenodoxeiou , efoson exei vrehei

	int k = 0;

	string finder = to_string(id);     // id einai to id toou stoixeiou pou theloume na vroume. To metatrepoume se string , dioti to arxeio mas exei strings.

	for (vector<vector<string>>::iterator it = vec_v.begin(); it != vec_v.end(); ++it) {    // grammiki anazitisi

		if ((*it).front() == finder)

		{

			cout << "hotel was found!" << endl;

			cout << "Here are the details of the hotel:" << endl;

			for (int i = 0; i < 4; ++i) {

				cout << "vector [" << k << "]" << "  " << (*it)[i] << endl;

			}

			break;		// an vroume to stoixeio den xreiazetai na psaksoume peretero, ara kanoume break

		}

		counterSearch++;	// counter sigkrisewn sto linear psaksimo

		k++;

	}

	if (k >= vec_v.size() - 1 ) {	// sinthiki gia to an vretike to arxeio i oxi

		cout << " to stoixeio den vrethike ..." << endl;

	}


}

void search_display_initialization(int id , ifstream& file, string& fname) {  // vriskei an iparxei to ksenodoxeio me to id pou dinoume ws orisma

	if (file.is_open() == 0 ) { file.open(fname); }    // antistoixi diadikasia me parapanw gia na perasw o arxeio sti mnimi

	string line;
```

```
int size;
string temp;
int moder = 0;
stringstream lineStream(line);
string cell;
while (getline(file, line))
{
	std::stringstream  lineStream(line);
	std::string	cell;
	while (std::getline(lineStream, cell, ';'))
	{
		temp = cell;
		break;
	}
	break;
}

vector<vector<string>> vec_v;
int l = 0;
while (getline(file, line)) {
	vector<string> v;
	std::stringstream  lineStream(line);
	std::string	cell;
	while (getline(lineStream, cell, ';'))
	{

		v.push_back(cell);

	}
	vec_v.push_back(v);
}

search_display_only(vec_v,id);    // kaloume tin parapanw sinartisi  pou kanei grammiki anazitisi
```

```
	cout <<
"================================================================================" << endl;


	if (file.is_open()) {
		file.close();
	}


	}

void display_by_surname( string surname,ifstream& file, string& fname) {  // pairnei ws orisma ena string, surname, kai etsi kai iparxei sto arxeio, ektipwnei ta stoixeia pou einai sxetiko me auto
	if (file.is_open() == 0 ) {
		file.open(fname);
	}

	clock_t begins = clock();  // ksekinaw metrisi pou diarkei to sorting, kai tin afairw apo to xrono ektelesis tou lagorithmou

	string line;   // paroimioia diadikasia me panw
	int size;
	string temp;
	int moder = 0;
	stringstream lineStream(line);
	string cell;
	while (getline(file, line))
	{
		std::stringstream  lineStream(line);
		std::string       cell;
		while (std::getline(lineStream, cell, ';'))
		{
			temp = cell;
			break;
		}
```

```
            break;
        }
        size = stoi(temp);

        vector<vector<string>> vec_v;
        int l = 0;
        while (getline(file, line)) {
            vector<string> v;
            std::stringstream  lineStream(line);
            std::string       cell;
            while (getline(lineStream, cell, ';'))
            {

                v.push_back(cell);

            }
            vec_v.push_back(v);
        }

        clock_t ends = clock();
        liniear_sur_init = double(ends - begins) / (CLOCKS_PER_SEC / 1000);  // den metrame ston teliko xrono, to xrono tou sortarismatos


        cout << "============================== display by surname ================================" << endl;
        int k = 0;
        for (vector<vector<string>>::iterator it = vec_v.begin(); it != vec_v.end(); ++it) {    // linear anazitisi me vasi ta epwnimata

            for (int i = 4; i <= ((*it).size() -1) ; i= i+3 ) {
                liniear_sur_comp++;
                if ((*it)[i] == surname) {
                    cout << "vrethike to epwnimo pou zitisate" << endl;
```

```
                    for (int printer = 0; printer < 3; ++printer) {
                        cout << (*it)[i+printer]<<endl;
                    }
                    break;
                }
                //cout << "vector [" << k << "]" <<  "  " << (*it)[i] << endl;
            }

            k++;
        }
        if (file.is_open() == 1) {
            file.close();
        }
}

vector<vector<string>> file_to_vector(ifstream& file, string& fname) {  // pairnei ws orisma ena arxeio, kai to pernaei se ena vector, wste na mporoume na to epeksergastoume
                                    // den pernaei tin prwti grammi

        if(file.is_open() == 0 ){  //anoigei to arxeio wste na paroume ta stoixeia
            file.open(fname);
        }


        string line;
        stringstream lineStream(line);
        string cell;
        vector<vector<string>> vec_v;
        int moder = 0;
        while (getline(file, line)) {

            if (moder == 0) {
                moder++;
```

```
                continue;
            }

            vector<string> v;
            std::stringstream  lineStream(line);
            std::string       cell;
            while (getline(lineStream, cell, ';'))  //prospelavnetai to arxeio, me stoixeia ana grammi
            {

                v.push_back(cell);
            }
            vec_v.push_back(v);
        }

        if (file.is_open() == 1) {
            file.close();   // kleinei to arxeio
        }

        return vec_v;   // epistrefei ton vector
}

vector<vector<string>> file_to_vector2(ifstream& file, string& fname) {  // pairnei ws orisma ena arxeio, kai to pernaei se ena vector, wste na mporoume na to epeksergastoume

                                                    // i diafora tou apo to apo panw, einai oti pernaei olo to vector, mazi kai tin prwti grammi

        if (file.is_open() == 0) {   //anoigei to arxeio wste na paroume ta stoixeia
            file.open(fname);
        }


        string line;
        stringstream lineStream(line);
```

```
	string cell;
	vector<vector<string>> vec_v;
	while (getline(file, line)) {

		vector<string> v;
		std::stringstream  lineStream(line);
		std::string       cell;
		while (getline(lineStream, cell, ';'))  //prospelavnetai to arxeio, me stoixeia ana grammi
		{

			v.push_back(cell);
		}
		vec_v.push_back(v);
	}

	if (file.is_open() == 1) {
		file.close();  // kleinei to arxeio
	}

	return vec_v;
}

bool basic_binary(string id, vector<vector<string>>& vec_v, string &start_s, string &end_s, string& mid_s , int end , int mid , int start ) {  //binary search to vasiko simeio

	while (start<=end) {     // i klasiki diadikasia gia to binary search .
		counterBinary++;    // metavliti pou metraei ton arithmo twn sigkrisewn
		if (id == mid_s) {  // se auti tin periptwsi exoume vrei to stoixeio, kai epitrefoumr timi true
			return true;
		}
		else if ( stoi(id) < stoi(mid_s) ) {    // metatrepoume to string id se int gia na ginei i sigkrisi.
			end = mid - 1;
			end_s = vec_v[end][0];
```

```
                mid = (start + end) / 2;

                mid_s = vec_v[mid][0];

                basic_binary(id, vec_v, start_s, end_s, mid_s, end, mid, start) ;  // afou id!=mid_s , me recursion ekteloume ksana tin diadikasia

        }

        else if ( stoi(id) > stoi(mid_s) ) {    //antistoixa me panw sxolio

                start = mid + 1;

                start_s = vec_v[start][0];

                mid = (start + end) / 2;

                mid_s = vec_v[mid][0];

                basic_binary(id, vec_v, start_s, end_s, mid_s, end, mid, start);

        }

    }


    return 0;   // an teleiwsei i while kai den exei ginei return true, tote paei na pei oti den vrethike to stoixeio, ara epistrefoume 0.


}


bool basic_interpolation(string id, vector<vector<string>>& vec_v, int& start, int& end, int& mid , int& e) {  // o vasikos kormos gia to interpolation search , einai recursive function


    string start_s = vec_v[start][0];   // ta idnexes twn stoixeiwn

    string end_s = vec_v[end][0];

    string mid_s = vec_v[mid][0];

    interpolSearch++;   // counter gia tis sigkriseis


    if ( (start <= end) && (e>= stoi(vec_v[start][0])) && (e<=stoi(vec_v[end][0]) )  ) {  // klasiki diadikasi interpolation search


        if (id == mid_s) {  // an to id = mid_s tote to stoixeio vrethike kai epistrefoume true

                cout << "to stoixeio: " << id << " vrethike!" << endl;

                return true;

        }
```

```
		else if (stoi(id) < stoi(mid_s) ) {

			end = mid - 1;

			end_s = vec_v[end][0];

			if (stoi(vec_v[end][0]) == stoi(vec_v[start][0])) {   // an isxiei auti i sigkrisi tote kseroume oti to stoixeio pou psaxnoume dn iparxei

				cout << " to stoixeio den vrethike ... " << endl;

				return 0;

			}		// an dn isxiei i parapanw id, tote allazoume tis times twn metavlitwn mas, kai sinexizoume me recursion

			mid = start + (((end - start)*(e - stoi(vec_v[start][0]))) / (stoi(vec_v[end][0]) - stoi(vec_v[start][0])));

			mid_s = vec_v[mid][0];


			basic_interpolation(id, vec_v, start, end, mid,e);   // recursion

		}

		else if (stoi(id) > stoi(mid_s) ) {    // antisoixa me panw


			start = mid + 1;

			start_s = vec_v[start][0];

			if (stoi(vec_v[end][0]) == stoi(vec_v[start][0]) ) {

				cout << " to stoixeio den vrethike... " << endl;

				return 0;

			}

			mid = start + ( ((end - start)*(e - stoi(vec_v[start][0]))) / ( stoi(vec_v[end][0]) - stoi(vec_v[start][0])) );

			mid_s = vec_v[mid][0];


			basic_interpolation(id, vec_v, start, end, mid, e);

		}

	}

	else {

		cout << "to stoixeio: " << id << " den vrethike ..." << endl;

		return 0;

	}
```

```
}

bool sorting(vector<string> v, vector<string> w) {  // sorting sinartisi montarismeni gia na doulevei me string stoixeia twn vectors

	return stoi(v[0])<stoi(w[0]);   // ta metatrepoume se ints wste na ginei swsta i sigkrisi

}

void binary_search_only(vector<vector<string>> vec_v, string id) {  // deutero stadio gia proetoimasia tou bianry search.
	int start = 0;			// index prwtou stoixeiou
	int end = vec_v.size() - 1;	// index teleutaioi
	int mid = (start + end) / 2;	// index mesaiou
	string start_s = vec_v[start][0];   // timi stoixeiou
	string end_s = vec_v[end][0];
	string mid_s = vec_v[mid][0];

	if (basic_binary(id, vec_v, start_s, end_s, mid_s, end, mid, start) == 1) { cout << "to stoixeio :" << id << " vrethike!" << endl;}  // ektipwnetai an exei vrethei i oxi to stoixeio pou zitisame
	else { cout << "to stoixeio :" << id << " den  vrethike ..." << endl; }
}

void binary_search(string id , ifstream& file, string& fname) {   // i proetoimasia gia to biary search

	cout << "============= BINARY SEARCH ===============  "  << endl;
	if (file.is_open() == 0) {
		file.open(fname);
	}
	vector<vector<string>> vec_v;
	vec_v = file_to_vector(file,fname);   // pername ta stoixeia se vector gia na perastoun stin mnimi kai na a epeksergastoume
```

```
	int moder = 0;



	clock_t begins = clock();  // ksekinaw metrisi pou diarkei to sorting, kai tin afairw apo to xrono ektelesis tou lagorithmou

	std::sort(vec_v.begin(), vec_v.end(), sorting);  // sort mesw tou sort tis stl

	clock_t ends = clock();

	binary_sorting = double(ends - begins) / (CLOCKS_PER_SEC / 1000);  // den metrame ston teliko xrono, to xrono tou sortarismatos



	binary_search_only(vec_v,id);   // ginetai i anazitisi , kalwntas tin parapanw sinartisi


	if (file.is_open() == 1 ) {

		file.close();

	}

}


void interpolation_search_only(vector<vector<string>> vec_v, string id) {  // arxikopoiisi twwn stoixeiwn gia to interpolation search

	int start = 0;

	int end = vec_v.size() - 1;

	int e = stoi(id);

	int mid = start + (((end - start)*(e - stoi(vec_v[start][0]))) / (stoi(vec_v[end][0]) - stoi(vec_v[start][0])));



	if (mid <= (stoi(vec_v[end][0]))) {

		basic_interpolation(id, vec_v, start, end, mid, e);

	}


	else cout << "to stoixeio :" << id << "den iparxei sti lista" << endl;

}


void interpolation_search(string id, ifstream& file, string& fname) {  // proetoimasia gia interpolation search
```

```
	cout << "============= INTERPOLATION SEARCH =============== " << endl;
	if (file.is_open()==0) {
		file.open(fname);
	}
	vector<vector<string>> vec_v;
	vec_v = file_to_vector(file, fname);
	int moder = 0;

	clock_t begini = clock();
	sort(vec_v.begin(), vec_v.end() , sorting);
	clock_t endi = clock();

	inter_sorting = double(endi - begini) / (CLOCKS_PER_SEC / 1000);  // den metrame to sorting ston xrono ektelesis tou algorithmou

	interpolation_search_only(vec_v , id);

	file.close();
}

//////////////////////////////////////////////  AVL DENTRO
////////////////////////////////////////////////////////////////////

	struct Node  // morfi node gia to AVL search
	{
		int key;
		vector<string>* v;
		struct Node *left;
		struct Node *right;
		int height;
	};
```

```
int height(struct Node *N)  // epistrefei to ipsos tou dentrou
{
        if (N == NULL)
                return 0;
        return N->height;
}

int max(int a, int b)  // Epistrefei to megisto metaksi 2 integers
{
        return (a > b) ? a : b;
}

 Node* newNode(int key, vector<string>& v)   // ftiaxnei ena neo node me kleidi pou theloume, kai arxikopoiei
{                                   // tous left kai right poiinters me NULL

         struct Node* node = new Node;  // desmevetai xwtos gia to neo stoixeio

        node->key = stoi(v[0]);  // to key pairnei tin timi pou theloume
        node->v = &v;
        node->left = NULL;
        node->right = NULL;

        node->height = 1;  // to neo node einai leaf tou dentrou
        return(node);  // epistrofi tou node
}

 Node *rightRotate(struct Node *y)       // kanei deksio rotation sto dentro me root to node y pou tou dinoume
{
        struct Node *x = y->left;
        struct Node *T2 = x->right;

        // edw ginetai to rotation
```

```
        x->right = y;
        y->left = T2;

        // ftiaxnoume ta nea ipsoi
        y->height = max(height(y->left), height(y->right)) + 1;
        x->height = max(height(x->left), height(x->right)) + 1;

        // epistrofi tis neas rizas
        return x;
}


 Node *leftRotate(struct Node *x)   // ginetai aristoro rotation me vasi to node x
{
        struct Node *y = x->right;
        struct Node *T2 = y->left;

        // edw ginetai to rotation
        y->left = x;
        x->right = T2;

        //  ftiaxnoume ta nea ipsoi
        x->height = max(height(x->left), height(x->right)) + 1;
        y->height = max(height(y->left), height(y->right)) + 1;

        // epistrefoume ti riza
        return y;
}


int getBalance(struct Node *N) //pairnoume to balance tou node N me vasi ta ipsoi
{
        if (N == NULL)
```

```
            return 0;
        return height(N->left) - height(N->right);
    }


    struct Node* insert(struct Node* node, vector<string>& v) // recursive function pou prosthetei to key se ipodentro me riza node kai epistrefei to nea riza tou subtree
    {
        int key = stoi(v[0]);

        // 1 . BST insertion
        if (node == NULL)
            return(newNode(key, v));

        if (key < node->key)
            node->left = insert(node->left, v);
        else if (key > node->key)
            node->right = insert(node->right, v);
        else                    //den epitrepontai sto BST idia kleidia
            return node;

        // 2.  Ginetai update tou ipsous tou progonou
        node->height = 1 + max(height(node->left),
            height(node->right));

        // 3.  Tsekaroume to balance tou progonou tou node gia na doume an exei ginei unbalanced
        int balance = getBalance(node);

        // An to node pou eidame prin , ginei unbalanced , tote exoume tis parakatww 4 periptwseis.

        // Left Left
        if (balance > 1 && key < node->left->key)
            return rightRotate(node);
```

```
        // Right Right
        if (balance < -1 && key > node->right->key)
                return leftRotate(node);

        // Left Right
        if (balance > 1 && key > node->left->key)
        {
                node->left = leftRotate(node->left);
                return rightRotate(node);
        }

        // Right Left
        if (balance < -1 && key < node->right->key)
        {
                node->right = rightRotate(node->right);
                return leftRotate(node);
        }

        return node;
    }

    void preOrder(struct Node *root, vector<vector<string>>& vec_v)  // ektipwsi tou preorder traversal (kiriws gia testing) kai tou ipsous kathe node
    {

        if (root != NULL)
        {
                //cout << root->key << "   name : " << (*(root->v))[1] << endl;                    /// maybe i want this
                preOrder(root->left, vec_v);
                preOrder(root->right, vec_v);

        }
```

```
}

bool search_avl(struct Node *root , string id , vector<vector<string>>& AVLvec_v ) {  // ginetai to avl search

        //AVLsearch++;   // counter gia sigkriseis sto avl tree

        if (root != NULL)
        {
                if (id == (*(root->v))[0] ) {
                        return 1;
                }
                if (search_avl(root->left, id, AVLvec_v) == 1) { AVLsearch++; return 1; };
                if(search_avl(root->right, id, AVLvec_v)==1) { AVLsearch++; return 1; };

        }

}

void AVL(ifstream& file, string& fname, string id) {  // proetoimasia gia to AVL search

        if (file.is_open()==0) {
                file.open(fname);
        }
        clock_t begin3 = clock();  // ksekinima metriseis gia arxikoopoiisi algorithmou, kai afairesi autou tou xrono apo to sinoliko xrono, wste na vrethei o xronos ektelesis
                        // tou algorithmou mono

        struct Node *root = NULL;
        vector <vector <string >>  AVLvec_v = file_to_vector(file, fname);  // antisoitxa me parapanw . Pername ta stoixeia tou file se vector.
        vector<string> v;
        for (int i = 0; i < AVLvec_v.size(); i++) {
```

```
                root = insert(root, AVLvec_v[i]);  // kaloume tin insert opws eipame parapanw
        }

        cout << "AVL anaparastasi :" << endl;    // ektipwsi tou avl dentrou (kiriws gia apodiksei oti leitourgei swsta)
        preOrder(root, AVLvec_v);           // gia na min argei i ektelesi sta peiramata exw valei se sxolia tin ektipwsi.

        clock_t end3 = clock();
        AVL_init = double(end3 - begin3) / (CLOCKS_PER_SEC/1000) ;  // den metrame to initialize time ston xrono ektelesis tou algorithmou

        if(search_avl(root, id, AVLvec_v) == 1) {
                cout << "to stoixeio me id : " << id<<"  vrethike!" << endl;
        }
        else { cout << " to stoixeio me id : " << id << "  den vrethike ..." << endl; }

        if (file.is_open() ) {
                file.close();
        }
    }

//////////////////////////////////////////          TRIE DENTRO
////////////////////////////////////////////////////////////////////////

#define ALPHABET_SIZE 26  // to megethos twn stoixeiwn tou leksikou

    struct node  // morfi node gia TRIE
    {
        vector<string> v;  // krataei ta stoixeia : onoma ksenodoxeiou , onoma pelati
        vector<string> k;   // krataei ta stoixeia tis kratisi me vasi to onoma tou pelati
        bool is_end;    // an einai leaf i oxi o kombos pou koitame
```

```
        string ksenodoxeio;
        string krtisi;
        int prefix_count;  // metraei tis lekseis sto leksiko pou exoun ena sigkrekrimeno string idi ws prefix
        struct node* child[ALPHABET_SIZE];  // pinakas diektwn pou deixnei se kathe pithano gramma tou alfavitou pou exoume
    } *head;


    void init()  // arxikopoiisi tou TRIE
    {
        head = new node();  // desmevoume xwro gia stoixeio tipou node
        head->prefix_count = 0;    // den exei kapoio prefix akoma, afou molis ftiaxtike
        head->is_end = false;   // an dn einai teleutaio gramma leksis einai false, alliws true. Edw ginetai initialization ara einai false
    }


    vector<node*> del; //mesw autwn ton pointers the eleftherwsw tin mnimi
    node* dill;

    void insert(vector<string>& v)   // prosthetei ti leksi pou tha tou dwsoume sto trie. Prosoxi, prepei na eleftherwnetai i mnimi meta tin ektelesi tou TRIE.
    {

        for (int i = 4; i < v.size(); i = i+3 ) {
            string word = v[i];    // diavazw ta stoixeia meta tin 4i thesi, ana 3 , wste na pairnw mono ta onomata

            node *current = head;

            current->prefix_count++;
            for ( int i = 0; i < word.length(); ++i)
            {
```

```
                int letter = (int)tolower(word[i]) - (int)'a';  // antistoixizei to gramma pou exoume se kapoioarithmo wste na vrethei to katalilo keli tou pinaka child
                                                                // kai na to simadepsoume

                if (current->child[letter] == NULL) {
                    current->child[letter] = new node();  // nei child sto dentro
                    dill = current->child[letter];  // mesw tou dill , tha valw tous pointers pou dimiourgountai se ena vector wste na mporesw
                    del.push_back(dill);          // na prospelasw auton ton vector meta , kai na eleftherwsw tin mnimi
                }

                current->child[letter]->prefix_count++;
                current = current->child[letter];

            }

            current->ksenodoxeio = v[1];  // eisagw ta stoixeia tou ksenodoxeiou gia to omoma pou psaxnw sto vector prwti fora
            current->v.push_back(v[1]);  //  eisagw ta sotixeia tis kratisis gia to onoma pou psaxnw  gia tin prwti fora

            for (int j = i ; j < i+3 ; j++) {
                current->k.push_back(v[j]);          // eisagw ta stoixeia tis kratisis
            }

            current->is_end = true;  // exoume teliki leksi
        }

    }

    int c = 0;

    bool search(string word)  // psaxnw ti leksi mesa sto trie
    {
```

```
node *current = head;
for ( int i = 0; i<word.length(); ++i)
{
	TrieSearch++;  // counter gia anazitiseis sto trie

	if (current->child[((int)word[i] - (int)'a')] == NULL)
	{

		return false;   // an den iparxei i leksi pou psaxnoume ws teliki
	}
	else
	{
		current = current->child[((int)word[i] - (int)'a')];  // sinexizoume to psaksimo

	}
	}

	clock_t begin1 = clock();   // ksekinaw metrisi, kai afairw ton xrono ektipwsis tou trie , apo ton teliko xrono, afinontas etsi mono to xrono ektelesis

	for (int k = 0; k < current->v.size(); k++) {
		cout << " onoma ksenodoxeiou pou exoun pelati me onoma  '"<< word  << "'  :" ;  // ektipwsi stoixeiwn
		cout << " " << current->v[k] << endl;

	}

	for (int l = 0; l < current->k.size(); l++) {
		cout << " stoixeia kratisis pelati me onoma  '"<<word <<"'  :" ;
		cout << "  " << current->k[l] << endl;


		if (c == 2) {
```

```
                cout << "----------------------------------" << endl;
                c = -1;
            }
            c++;
        }

        clock_t end1 = clock();
         Trie_init2 = double(end1 - begin1) / (CLOCKS_PER_SEC / 1000);  // den metraw to initialization time ston teliko xrono ektelesis algorithmou


        return current->is_end; // 0 an den tin vrika,  1 ama ti vrika ti leksi pou epsaxna
    }

    void trie(ifstream& file, string& fname ,string surname) {   // arxikopoiisi kai ektelesi tou TRIE.
        cout << "===================================== TRIE   SEARCH ================" << endl;

        if (file.is_open()==0) {
            file.open(fname);
        }

        clock_t begin1 = clock();   // ksekinaw metrisi, kai afairw ton xrono tis arxikopoiisis tou trie , apo ton teliko xrono, afinontas etsi mono to xrono ektelesis

        vector <vector <string >>  vec_v = file_to_vector(file, fname); // vector pou kratei ta stoixeia tou arxeiou

        init();

        for (int i = 0; i<vec_v.size(); ++i)
        {
            insert(vec_v[i]);   // vazw sto trie ta stoixeia tou vector pou exei to arxeio
```

```
                }

                string s = surname;

                boost::algorithm::to_lower(s);  // oti eisodo kai an paroume (string) tin metatrepoume se lower case

                clock_t end1 = clock();
                 Trie_init = double(end1 - begin1) / (CLOCKS_PER_SEC / 1000);  // den metraw to initialization time ston teliko xrono ektelesis algorithmou

                cout << search(s) << endl;  // anazisi sto initialized trie

                for (vector<node*>::iterator it = del.begin(); it != del.end(); it++ ) {   // eleftherwnw tin mnimi apo ta pointers pou dimiourgithikan pio prin
                        delete(*it);
                }
                del.clear();   // adeiazw ton vector pou kratouse tous pointers pou dimiourgithikan prin


                if (file.is_open()==1) {
                        file.close();
                }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

        vector<string> get_all_names(ifstream& file, string& fname) {  // pairnei mono ta onomata apo to file, ta vazei se ena vector, kai  epistrefei to vector
                if (file.is_open() == 0) {
                        file.open(fname);
                }
```

```
vector<string> v;
string line;
stringstream lineStream(line);
string cell;
vector<vector<string>> vec_v;
int moder = 0;

while (getline(file, line)) {
	int counter = 0;
	int counter2 = 0;
	if (moder == 0) {
		moder++;
		continue;
	}

	std::stringstream  lineStream(line);
	std::string	cell;
	while (getline(lineStream, cell, ';'))
	{
		if (counter <4) {
			counter++;
		}
		else {
			if (counter2 == 3) {
				v.push_back(cell);
			}
			counter2++;
		}
	}
}

if (file.is_open() == 1) {
	file.close();
```

```
        }
        cout << " returining v with size : " << v.size() << endl;
        return v;
    }

    string get_rand_name(vector<string>& vec) {  // epistrefei random name, apo vector me onomata pou tou dinoume ws orisma
        int randn = rand() % (vec.size() - 1) +1;  // otan o counter ginei isos me ton randn , tha epistrafei to onoma sto opoio deixnei o iterator
        int counter = 0;
        for (vector<string>::iterator it = vec.begin(); it != vec.end(); it++) {  // kanw iterate ta onomata pou exw lavei apo tin proigoumeni sinartisi,
                                                                                  // kai me vasi to tandn pairnw ena random onoma opws fainetai parakatw
            if (counter == randn) {
                std::cout << "rand name generated : " << *it << endl;
                return *it;
            }

            counter++;
        }
    }

    void metriseis(ifstream& file, string& fname) {  //sinartisi pou kanei mazemenes oles tis metriseis . Einai gia testing me polles epanalipseis

        double avrg_linear = 0;   // xronoi ektelesis algorithmwn
        double avrg_binary = 0;
        double avrg_interpol = 0;
        double avrg_AVL = 0;
        double avrg_TRIE = 0;
        double linear_sur = 0;
```

```
if (file.is_open() == 0) {
        file.open(fname);
}
vector<vector<string>> vec_v = file_to_vector(file, fname);    // arxeio se vector
vector<vector<string>> vec_v2 = file_to_vector2(file, fname);
file.close();

vector<string> vec = get_all_names(file, fname);   // anaktisi olwn twn onomatwn, me skopo na ginei i anazitisi tixaiwn onomatwn pou iparxoun omws sto arxeio
srand(time(NULL));

for (int j = 0; j < EPANAL; j++) {  // orizoume poses fores theloume na ginoun oi epanalipseis gia tis metriseis mas

cout << "                                    j   " << j << endl;  // aplimetavliti pou mas enimerwnei poses metriseis exun ginei. (perito, apla voliko)

int i = rand() % stoi(vec_v2[0][0]) +1;  // arithmos pou kimenetai apo ti mikroteri mexri tin megaliteri timi id

string tester;
tester = get_rand_name(vec);  // tester pairnei ena random onoma

string is = to_string(i);    // tixaios string arithmos


clock_t begin = clock();    // ksekinaei o xrono metrisis gia kathe algorithmo , kai ipologizetai o mesos oros twn metrisewn .
search_display_initialization(i, file, fname);
clock_t end = clock();
double elapsed_secs = double(end - begin) / (CLOCKS_PER_SEC / 1000);
avrg_linear = avrg_linear + elapsed_secs;
```

```
clock_t begin1 = clock();
binary_search(is, file, fname);
clock_t end1 = clock();
double elapsed_secs1 = double(end1 - begin1) / (CLOCKS_PER_SEC / 1000) - binary_sorting;
avrg_binary = avrg_binary + elapsed_secs1;

clock_t begin2 = clock();
interpolation_search(is, file, fname);
clock_t end2 = clock();
double elapsed_secs2 = double(end2 - begin2) / (CLOCKS_PER_SEC / 1000) - inter_sorting;
avrg_interpol = avrg_interpol + elapsed_secs2;

clock_t begin3 = clock();
AVL(file, fname, is);
clock_t end3 = clock();
double elapsed_secs3 = double(end3 - begin3) / (CLOCKS_PER_SEC / 1000) - AVL_init;
avrg_AVL = avrg_AVL + elapsed_secs3;

clock_t begin4 = clock();
trie(file, fname, tester);
clock_t end4 = clock();
double elapsed_secs4 = double(end4 - begin4) / (CLOCKS_PER_SEC / 1000) - Trie_init -
Trie_init2;
avrg_TRIE = avrg_TRIE + elapsed_secs4;

clock_t begin5 = clock();
display_by_surname(tester, file, fname);
clock_t end5 = clock();
double elapsed_secs5 = double(end5 - begin5) / (CLOCKS_PER_SEC / 1000) - liniear_sur_init;
linear_sur = linear_sur + elapsed_secs5;
}

// efoson teleiwsei o arithmos epanalipsewn , exoume sileksei stoixeia gia xrono ektelesis kai
arithmo sigkrisewn, kai parousiazoume ta stoixeia mesw tou parakatw pinaka
```

```
		cout << endl << endl << "--------------------------     METRISEIS    ---------------------------------------" << endl << endl;

		cout << " serch_display execution time: " << avrg_linear/EPANAL << "   kai sigkriseis : " << counterSearch/EPANAL << endl;

		cout << " binary_sorting execution time : " << avrg_binary/ EPANAL << "   kai sigkriseis : " << counterBinary/EPANAL << endl;

		cout << " interpoltion_search execution time : " << avrg_interpol/EPANAL << "   kai sigkriseis : " << interpolSearch/EPANAL << endl ;

		cout << " AVL tree  execution time : " << avrg_AVL/EPANAL << "    kai sigkriseis : " << AVLsearch/EPANAL << endl ;

		cout << " Linear surname search execution time : " << linear_sur / EPANAL << "    kai sigkriseis : " << liniear_sur_comp / EPANAL << endl ;

		cout << " Trie execution time : " << avrg_TRIE/EPANAL <<"    kai sigkriseis : " << TrieSearch/EPANAL << endl << endl;

	}



	void save_hotels(ifstream& file, string& fname , ofstream& saved, string& fout) { //apothikvei ta ksekodoxeia se arxeio pou dexetai ws orisma


		if ( file.is_open()==0 ) {	// klasiki diadikasia metaforas apo arxeio se vector, kai apo vector se arxeio

			file.open(fname);

		}


		if (saved.is_open() == 0) {

			saved.open(fout);

		}


		vector<vector<string>> vec_v;

		vec_v = file_to_vector2(file, fname);  //pername ki tin prwti grammi


		int counter = 0;

		for (vector<vector<string>>::iterator it = vec_v.begin(); it != vec_v.end(); it++ ) {

			if (counter == 0) {
```

```
                saved << vec_v.size() << ";" << endl;   //enimerwnoume tin nea prwti grammi
            }
            else {

                for (int i = 0; i < (*it).size(); i++ ) {

                    if (i <  (*it).size() -1 ) {
                        saved << (*it)[i] << ";";
                    }
                    else {
                        saved << (*it)[i] << endl;

                    }
                }

            }

            counter++;
        }

        if (file.is_open() == 1) {
            file.close();
        }
        if (saved.is_open() == 1 ) {
            saved.close();
        }
    }

    ////////    OI PARAKATW 3 SINARTISEIS , EINAI GIA MEMONOMENO TESTING KATHE PERIPTWSIS. OMWS EPEIDI EXW TIN SINARTISI metriseis DEN EINAI APARAITITES PLEON

    double linear_search_metriseis(ifstream& file, string& fname) {   // xrisimopoieitai an theloume na kanoume metriseis mono gia ton linear tropo prospelasis
```

```
        double elapsed_secs = 0;

        if (file.is_open() == 0) {
                file.open(fname);
        }
        vector<vector<string>> vec_v = file_to_vector(file, fname);
        file.close();
        int vec_size = vec_v.size();

        clock_t begin = clock();
        int i;
        for (int k = 0; k < EPANAL; k++ ) {
                 i = rand() % vec_size +1;
                string is = to_string(i);
                cout << "                                        k = " << k <<  "    i = "<< i <<endl;
                search_display_only(vec_v , i);
        }
        clock_t end = clock();
        elapsed_secs = double(end - begin) / (CLOCKS_PER_SEC/1000) ;

        cout << " stis 1000 metriseis ,mesos oros : " << elapsed_secs << endl;


        //return elapsed_secs / EPANAL;
        return elapsed_secs;
    }

    double binary_search_metriseis(ifstream& file, string& fname) {  // xrisimopoieitai an theloume na kanoume metriseis mono gia ton binary tropo prospelasis
        double elapsed_secs ;
        if (file.is_open() == 0) {
                file.open(fname);
```

```
		}
		vector<vector<string>> vec_v = file_to_vector(file, fname);
		file.close();
		int vec_size = vec_v.size();

		clock_t begin = clock();
		int i;
		for (int k = 0; k < EPANAL; k++) {
			i = rand() % vec_size +1;
			string is = to_string(i);
			cout << "                                    k = " << k << "    i = " << i << endl;

			binary_search_only(vec_v , is);
		}
		clock_t end = clock();
		elapsed_secs = double(end - begin) / (CLOCKS_PER_SEC/ 1000) ;

		cout << " stis 1000 metriseis ,mesos oros : " << elapsed_secs  << endl;

		return elapsed_secs ;
	}

	double interpolation_search_metriseis(ifstream& file, string& fname) {   // xrisimopoieitai an theloume na kanoume metriseis mono gia ton interpolation tropo prospelasis

		double elapsed_secs;
		if (file.is_open() == 0) {
			file.open(fname);
		}
		vector<vector<string>> vec_v = file_to_vector(file, fname);
		file.close();
		int vec_size = vec_v.size();
```

```
		clock_t begin = clock();
		int i;
		for (int k = 0; k < EPANAL; k++) {
			i = rand() % vec_size +1 ;
			string is = to_string(i);
			cout << "                                    k = " << k << "    i = " << i << endl;

			interpolation_search_only(vec_v, is);
		}
		clock_t end = clock();
		elapsed_secs = double(end - begin) / (CLOCKS_PER_SEC/1000 ) ;

		cout << " stis 1000 metriseis ,mesos oros : " << elapsed_secs  << endl;
		return elapsed_secs ;
	}

	/////////


int main(int argc, const char * argv[])
{

	string fname;
	if (argv[1] != NULL ) {   // dinatotia epilogis onomatos mesw terminal me argv . An den dextei onoma arxeio me auto ton tropo, tote dinetai default onoma
		fname = argv[1];
	}
	else {
		 fname = "data.csv";   // default onoma
	}
	string fout = "new.csv";   // default onoma gia apothikeusi se allo arxeio
	ifstream file(fname);
	ofstream out(fname, ios::app);
```

```
ofstream saved;



while (1) {                    // kormos ektelesis programmatos .
        ifstream file(fname);
        ofstream out(fname, ios::app);
        ofstream saved;

        cout << endl << endl << "======================   MENU
============================  " << endl << endl;
        cout << " 1. Load Hotels and Reservations from file ." << endl;
        cout << " 2. Save Hotels and Reservarions to file ." << endl;
        cout << " 3. Add Hotel (and it's reservations) ." << endl;
        cout << " 4. Search and Display Hotel by id ." << endl;
        cout << " 5. Display Reservations by surname search ." << endl;
        cout << " 6. Metriseis gia testing ." << endl;
        cout << " 7. Exit ." << endl;

        int chooser;
        cin >> chooser;
        switch (chooser) {
        case 1: { load_hotels(file, fname); } break;
        case 2: {save_hotels(file, fname, saved , fout);  }   break;
        case 3: { add_hotel(file, out, fname); } break;
        case 4:
        {
                cout << " 1. Linear Search ." << endl;
                cout << " 2. Binary Search ." << endl;
                cout << " 3. Interpolation Search ." << endl;
                cout << " 4. AVL tree  ." << endl;
                int chooser2;
                cin >> chooser2;
                if (chooser2 == 1) {
```

```
                cout << " Give me the ID you want to search :" << endl;
                int id;
                cin >> id;
                search_display_initialization(id, file, fname);
        }
        else if (chooser2 == 2) {
                cout << " Give me the ID you want to search :" << endl;
                string id;
                cin >> id;
                binary_search(id, file, fname);
        }
        else if (chooser2 == 3) {
                cout << " Give me the ID tou want to search : " << endl;
                string id;
                cin >> id;
                interpolation_search(id, file, fname);
        }
        else if (chooser2 == 4) {
                cout << " Give me the ID tou want to search : " << endl;
                string id;
                cin >> id;
                AVL(file, fname, id);
        }
}
        break;

case 5:
{
        cout << " 1. Linear Search." << endl;
        cout << " 2. Trie ." << endl;
        int ch;
        cin >> ch;
        if (ch == 1) {
```

```
                cout << " Give me the SURNAME tou want to search : " << endl;
                string surnamee;
                cin >> surnamee;
                display_by_surname(surnamee, file, fname);
            }
            else {
                cout << " Give me the SURNAME tou want to search : " << endl;
                string surnamee;
                cin >> surnamee;
                trie(file , fname , surnamee );
            }

            }
            break;

        case 6: {
            metriseis(file, fname);
             counterBinary = 0;
             counterSearch = 0;
             interpolSearch = 0;
          AVLsearch = 0;
             TrieSearch = 0;
             liniear_sur_comp = 0;
        } break;

        case 7: exit(0); break;
        }

    }


    return 0;
}
```