Ονοματεπώνυμο: *Θεοφανόπουλος Μιχαήλ*
Αριθμός Μητρώου: *1115201800053*
1η Εργασία Λειτουργικά Συστήματα

➔ *Αρχικά, η εργασία έχει υλοποιηθεί σε C++. Επέλεξα τη συγκεκριμένη γλώσσα καθώς είμαι περισσότερο εξοικειωμένος σε αυτή και επίσης θεώρησα χρήσιμες τις αντικειμενοστραφείς δυνατότητες της καθώς και την ύπαρξη constructors/destructors με σκοπό να είναι πιο απλή η χρήση των δομών που αναφέρονται στην εκφώνηση και η αποδέσμευση του χώρου κατά τον τερματισμό του προγράμματος (κυρίως όπου είναι εφικτή και προτιμάται η μη δυναμική δέσμευση μνήμης για τα αντικείμενα που θα αντιπροσωπεύουν τις δομές). Η εργασία μου χρησιμοποιεί βασικές βιβλιοθήκες, όπως:*

1) *την <iostream> με σκοπό να παρέχεται η δυνατότητα να διαβάζονται δεδομένα από τον χρήστη καθώς και να εκτπώνονται κατάλληλα μηνύματα στον χρήστη,*
2) *την <fstream> με σκοπό να παρέχεται η δυνατότητα να διαβάζει το πρόγραμμα δεδομένα από text files (κυρίως για το κομμάτι που ορίζει η εκφώνηση πώς πρέπει το πρόγραμμα να διαβάζει δεδομένα από αρχείο που δίνεται κατά την εκτέλεση του προγράμματος και με τα δεδομένα αυτά να ενημερώνει τις δομές, καθώς και μια συνάρτηση η οποία επιστρέφει τον αριθμό δεδομένων που περιέχει ένα αρχείο και για την οποία θα αναφερθούμε παρακάτω)*
3) *την <string> με σκοπό να παρέχεται η δυνατότητα να χρησιμοποιούνται στο πρόγραμμα δεδομένα τύπου string, καθώς και ορισμένες ιδιαίτερες χρήσιμες συναρτήσεις του συγκεκριμένου τύπου,*

4) *την <time.h> με σκοπό να παρέχεται η δυνατότητα να δέχεται το πρόγραμμα τη τρέχουσα ημερομηνία κατευθείαν απο το λειτουργικό σύστημα*
5) *την <string.h> με σκοπό να παρέχεται η δυνατότητα χρήσης της συνάρτησης strcmp().*

➔ *Η εργασία αποτελείται από 7 αρχεία κώδικα και ένα makefile. Όλα τα δεδομένα όπως typedef , global variables κλπ βρίσκονται στο αρχείο types.hpp. Στο συγκεκριμένο αρχείο επίσης δηλώνονται και όλες οι κλάσεις με τα χαρακτηριστικά τους και τις συναρτήσεις τους. Τα σώματα των συναρτήσεων της κάθε κλάσης είναι γραμμένα στο αντίστοιχο εκτελέσιμο αρχείο που υπάρχει για την κάθε κλάση. Για παράδειγμα το σώμα του constructor της κλάσης HashNode είναι γραμμένο στο αρχείο HashNode.cpp. Εκτός των παραπάνω, υπάρχει και το αρχείο functions.cpp στο οποίο περιγράφονται όλες οι συναρτήσεις που χρησιμοποιούνται στο πρόγραμμα και δεν αφορούν κάποια κλάση συγκεκριμένα, είναι δηλαδή γενικού σκοπού. Ο λόγος που επέλεξα τη συγκεκριμένη μορφοποίηση για τα αρχεία του προγράμματος μου είναι διότι θεώρησα εύχρηστο όλες οι κλάσεις να δηλώνονται στο ίδιο αρχείο για να είναι πιο εύκολο οποιοσδήποτε να τις ανατρέξει, ενώ οι συναρτήσεις κάθε κλάσης επέλεξα να βρίσκονται σε ξεχωριστό αρχείο διότι είναι πιο εύκολο για τον αναγνώστη να ανατρέξει σε μια συγκεκριμένη συνάρτηση μίας συγκεκριμένης κλάσης από το να υπήρχε όλος ο εκτελέσιμος κώδικας μαζεμένος σε μοναδικό αρχείο. Τέλος, υπάρχει το αρχείο mngstd.cpp στο οποίο ορίζεται η main και υλοποιείται το ζητούμενο της εργασίας. Το makefile κάνει χρήση separate compilation με σκοπό της μεταγλώττιση όλου του προγράμματος και την χρήση της εντολής ~$ make all. Επίσης μέσα στο makefile, περιγράφονται διάφορες άλλες δυνατότητες. Για εκτέλεση του προγράμματος πρέπει να δωθεί η εντολή ~$ ./mngstd -i texfile.txt. Υπάρχει η δυνατότητα να εκτελεστεί το πρόγραμμα χωρίς input file, αλλά στη περίπτωση αυτή το μέγεθος του*

hash table αρχικοποιείται εξ αρχής σε 20 θέσεις. Για παραπάνω δεδομένα δεν εγγυόμαι τη σωστή λειτουργία του. Σε περίπτωση που δωθεί input file, το μέγεθος του πίνακα ορίζεται εξ αρχής ίσο με τον αριθμό των δεδομένων που περιέχει το text file.

- *Κατά την εκκίνηση του προγράμματος το πρόγραμμα τρέχει τη συνάρτηση getCurrentYear() του αρχείου functions.cpp με σκοπό να πάρει το current year και να το αποθηκεύσει στη global μεταβλητή currentYear. Στη συνέχεια, ελέγχει εάν ο χρήστης έχει δώσει input file κατά την εκτέλεση του προγράμματος και αν ναι βρίσκει τον αριθμό δεδομένων που περιέχονται στο αρχείο και αποθηκεύει τον αριθμό αυτόν στη global μεταβλητή TABLESIZE. Εάν ο χρήστης δεν έχει δώσει input file, τότε το TABLESIZE αρχικοποιείται με 20, όπως και αναφέρθηκε προηγουμένως. Ύστερα, ελέγχει ξανά αν έχει δώσει ο χρήστης input file και αν ναι εισάγει στις δύο δομές που δημιουργεί στατικά (HashTable ht,InvertedIndex) τα δεδομένα που περιέχονται στο αρχείο χρησιμοποιώντας τη συναρτήση fill η οποία επίσης περιγράφεται στο αρχείο fun*

  Τέλος, δίνει επαναληπτικά στον χρήστη τη δυνατότητα να ε εντολή σχετικά με το τι ενέργεια θέλει να κάνει, μέχρι ο χρήστης να δώσει την εντολή 10 και όπου το πρόγραμμα θα τερματίσει αφού πρώτα απελευθερωθεί κατάλληλα όλη μνήμη που έχει δεσμευτεί. Επέλεξα να ακολουθήσω τον συγκεκριμένο τρόπο υποδοχής δεδομένων από τον χρήστη καθώς ελαχιστοποιεί τα false inputs και θεωρώ πως είναι πιο κατανοητό και εύκολο στη χρήση.

- *Για τον σχεδιασμό των βάσεων που αναφέρονται στην εκφώνηση επέλεξα την εξής υλοποίηση:*

  i. *Η κλάση που αντιπροσωπεύει τους students και περιέχει όλα τα στοιχεία τους είναι η HashNode. Εκτός από τα*

στοιχεία του κάθε μαθητή υπάρχει ένας δείκτης HashNode next, καθώς αποτελεί λίστα από μαθητές.Η λίστα αυτή ανήκει στη*

ii. *κλάση HashTable. Η κλάση αυτή αντιπροσωπεύει το HashTable που αναφέρεται στην εκφώνηση. Ως μοναδικό χαρακτηριστικό έχει έναν δείκτη σε πίνακα από HashNodes. Τα HashNodes αυτά συνδέονται μεταξύ τους σε μορφή λίστας. Με λίγα λόγια έχουμε ένα HashTable το οποίο περιέχει σε κάθε κελί του μία λίστα απο HashNodes, δηλαδή μία λίστα από μαθητές. Η εισαγωγή των μαθητών στο HashTable γίνεται με βάση το StudentID τους και με τη χρήση της συνάρτησης HashFunction(), την οποία έχω υλοποιήσει με τέτοιο τρόπο ώστε να ελαχιστοποιούνται όσο το δυνατόν περισσότερο τα collisions. Το μέγεθος του HashTable αρχικοποιείται εξ αρχής με τη χρήση της getFileSize() και αν δεν έχει δωθεί input file αρχικοποιείται με τη default τιμη 20,όπως αναφερθήκαμε προηγουμένως. Επέλεξα την συγκεκριμένη υλοποίηση, καθώς θεώρησα ότι ανταποκρίνεται βέλτιστα στις προδιαγραφές που ορίζονται στην εκφώνηση της άσκησης.*

iii. *Επίσης υπάρχει η κλάση InvertedIndex, η οποία έχει για χαρακτηριστικό έναν integer Y, έναν δείκτη InvertedIndex* next, καθώς πάλι αποτελεί λίστα από InvertedIndexes και τέλος έναν StudentList* student το οποίο στη πραγματικότητα αποτελεί μία nested λίστα από StudentList. Με λίγα λόγια έχουμε μια λίστα απο InvertedIndexes , δηλαδή έναν κόμβο για κάθε έτος εγγραφής στο πανεπιστήμο και για κάθε κόμβο έχουμε μια λίστα από μαθητές που είναι γραμμένοι το έτος αυτό στο πανεπιστήμιο.*

iv. *Η κλάση StudentList, έχει για χαρακτηριστικά έναν δείκτη StudentList* next αφού αποτελεί λίστα και έναν δείκτη HashNode* student. Ο λόγος που επέλεξα να περιέχει δείκτη σε HashNode και όχι να αποτελεί η*

συγκεκριμένη κλάση έναν μαθητή από μόνη της, είναι διότι η εκφώνηση της άσκησης ορίζει πως κάθε node της λίστας που περιλαμβάνεται σε ένα node του InvertedIndex πρέπει να δείχνει σε έναν μαθητή και όχι να αποτελεί έναν μαθητή, οπότε επέλεξα να έχει pointer σε HashNode. Όλες οι λίστες που ανέφερα παραπάνω αυξάνονται σε μέγεθος κάθε φορά που προστίθεται αντικείμενο το οποίο δεν υπάρχει ήδη στη λίστα.

v. *Όσον αφορά άλλες δομές που χρησιμοποιώ στο πρόγραμμα μου, σε ορισμένες συναρτήσεις (t(op),m(inimum)) χρησιμοποιώ τη κλάση StudentList, με σκοπό να έχω τη δυνατότητα να επιστρέφω αρκετούς μαθητές οι οποίοι πληρούν τις προδιαγραφές που ορίζει η εκφώνηση, δηλαδή όταν δεν είναι αναγκαστικό ότι πρέπει να επιστραφεί μοναδικός μαθητής. Επιπροσθέτως, σε κάποιες άλλες συναρτήσεις (t(op),p(ostal code)) χρησιμοποιώ στατικά δεσμευμένο πίνακα με σκοπό να αποθηκεύω εκεί τα κατάλληλα δεδομένα όπως ορίζουν οι προδιαγραφές της συνάρτησης και να μπορώ εύκολα να τα ταξινομήσω κατευθείαν και να τα εμφανίσω. Οι destructors των παραπάνω κλάσεων αποδεσμεύουν κατάλληλα όλη τη μνήμη που δεσμεύεται κατά την εκτέλεση του προγράμματος. Τέλος, ο λόγος που επέλεξα να αρχικοποιώ στατικά τα ii,ht στη main είναι διότι οι destructors των δύο αυτών κλάσεων θα κληθούν αυτόματα κατά τον τερματισμό του προγράμματος, μια δυνατότητα την οποία παρέχουν οι κλάσεις της C++ και επέλεξα να την αξιοποιήσω.*

➔ *Για τις συναρτήσεις διεπαφής του προγράμματος, η υλοποίηση είναι απλοϊκή και σε οποιοδήποτε σημείο θεώρησα ότι χρειάζεται να εξηγήσω τη λογική μου, υπάρχουν κατάλληλα σχόλια. Το πρόγραμμα έχει υλοποιήσει τη μορφή εξόδου ακριβώς όπως ορίστηκε στο αρχείο που υπάρχει στη σελίδα του μαθήματος(https://www.alexdelis.eu/k22/formatted-*

output.f20-prj1-v2.txt). Για κάθε συνάρτηση που έχω υλοποιήσει, η λειτουργία της έχει τη δυνατότητα να εκτελεστεί σε οποιαδήποτε από τις δύο δομές που υπάρχουν(HashTable,InvertedIndex), παρόλα αυτά by default το έχω ορίσει να εκτελεί την λειτουργία στην δομή στην οποία ο αλγόριθμος θα έχει καλύτερη απόδοση και θα είναι μικρότερο το κόστος προσπέλασης. Ο κώδικας που εκτελεί την ίδια λειτουργία στην άλλη δομή, δηλαδή στη δομή στην οποία έχω θεωρήσει πως θα είχε χειρότερη απόδοση βρίσκεται σε σχόλιο μέσα στο σώμα της κάθε συνάρτησης. Πιο συγκεκριμένα,

i. *Η i(nsert) δουλεύει πάνω και στις δύο δομές καθώς αργότερα στο πρόγραμμα πρέπει και οι δύο δομές να είναι κατάλληλα ενημερωμένες.*
ii. *Η l(ook-up) δουλεύει πάνω στον Inverted Index καθώς μπορούμε πιο γρήγορα να βρούμε το σωστό year και εκεί να εκτελέσουμε την αναζήτηση μας.*
iii. *Η d(elete) δουλεύει πάνω και στις δύο δομές καθώς αργότερα στο πρόγραμμα πρέπει και οι δύο δομές να είναι κατάλληλα ενημερωμένες.*
iv. *Η n(umber) δουλεύει πάνω στον Inverted Index καθώς μπορούμε πιο γρήγορα να βρούμε το σωστό year και εκεί να μετρήσουμε τον αριθμό των μαθητών που περιέχονται στη λίστα από students.*
v. *Η t(op) δουλεύει πάνω στον Inverted Index καθώς μπορούμε πιο γρήγορα να βρούμε το σωστό year και εκεί να κάνουμε τους κατάλληλους υπολογισμούς.*
vi. *Η a(verage) δουλεύει πάνω στον Inverted Index καθώς μπορούμε πιο γρήγορα να βρούμε το σωστό year και εκεί να κάνουμε τους κατάλληλους υπολογισμούς πάνω στη λίστα.*
vii. *Η m(inimum) δουλεύει πάνω στον Inverted Index καθώς μπορούμε πιο γρήγορα να βρούμε το σωστό year και εκεί*

να κάνουμε τους κατάλληλους υπολογισμούς πάνω στη λίστα.

viii. *Η c(ount) δουλεύει πάνω στον Inverted Index καθώς υπάρχει πιο εύκολη πρόσβαση σε κάθε year ξεχωριστά με σκοπό να υπολογιστούν τα κατάλληλα αποτελέσματα.*

ix. *Η p(ostal code) δουλεύει πάνω στον Hash Table, αλλά θα μπορούσε κάλλιστα να δουλεύει πάνω σε οποιαδήποτε δομή καθώς η πολυπλοκότητα παραμένει ίδια.*

➔ *Σε ορισμένα σημεία του προγράμματος, χρησιμοποιώ μικρές λειτουργίες για τις οποίες συμβουλεύτηκα κάποια πηγή από το ίντερνετ. Τα source σε αυτές τις περιπτώσεις αναφέρονται πάνω από τη συνάρτηση/κώδικα μέσα σε σχόλια. Πάντως, πολύ μικρές σε μέγεθος και σημαντικότητα λειτουργίες προέρχονται από τέτοιες πηγές, ενώ όλος ο υπόλοιπος κώδικας ανήκει αποκλειστικά σε εμένα.*

➔ *Τέλος, επέλεξα να μην συμπεριλαμβάνω στο πρόγραμμά μου τη δυνατότητα να δώσει ο χρήστης config file, καθώς θεώρησα πως δεν είναι ιδιαίτερα χρήσιμο για το πρόγραμμα μου και πως τα δεδομένα του μπορούν να συμπεριληφθούν στα κυρίως αρχεία χωρίς να χάνει το πρόγραμμα ιδιαίτερα σε οργάνωση.*

Σας εύχομαι καλή διόρθωση!