Μαστοράκη Αικατερίνα
AM: 1115201400100
Systems Programming 2020 - Project1

Όπου έχει χρησημοποιηθεί κώδικας, ο οποίος έχει βρεθεί στο internet επισημαίνεται ρητά ακριβώς πάνω απο την υλοποίηση. Το project1 του μαθήματος αποτελείται απο ένα σύνολο αρχείων πηγαίου και των αντίστοιχων header files που περιλαμβανουν τις δηλώσεις των συναρτήσεων και των δομών (definitions.h) που απαιτούνται. Τα αρχεία αυτά είναι:

- diseaseMonitor.c , το οποίο περιλαμβάνει τον βασικό κώδικα εκτέλεσης του project καθώς και τον κώδικα κλήσης των συναρτήσεων που υλοποιούν τις απαιτούμενες λειτουργίες του προγράμματος
- LinkedList.c, το οποίο περιλαμβάνει τις συναρτήσεις δημιουργίας, εισαγωγής, ανανέωσης, αναζήτησης και εκτύπωσης της λίστας. Στο αρχείο LinkedList.h υπάρχουν οι δηλώσεις των συναρτήσεων που υλοποιούνται στο αρχείο LinkedList.c.
- HashTable.c, το οποίο περιλαμβάνει τις συναρτήσεις δημιουργίας, εισαγωγής, ανανέωσης, αναζήτησης και εκτύπωσης των hashtables. Στο αρχείο HashTable.h υπάρχουν οι δηλώσεις των συναρτήσεων που υλοποιούνται στο αρχείο HashTable.c.
- SelfBalancedTree.c, όπου υλοποιείται το AVL BSStree. Στο αρχείο SelfBalancedTree.h υπάρχουν οι δηλώσεις των συναρτήσεων που υλοποιούνται στο αρχείο SelfBalancedTree.c.
- Heap.c, όπου υλοποιείται το maxHeap.. Στο αρχείο Heap.h υπάρχουν οι δηλώσεις των συναρτήσεων που υλοποιούνται στο αρχείο Heap.c.
- functions.c, όπου υπάρχουν διάφορες βοηθητικές συναρτήσεις που καλούνται απο τα παραπάνω αρχεία καθώς και οι συναρτήσεις που εξυπηρετούν τα ερωτήματα του χρήστη. Στο αρχείο functions.h υπάρχουν οι δηλώσεις των συναρτήσεων που υλοποιούνται στο αρχείο functions.c.

Η μεταγλώττιση του αρχείου πραγματοποιείται, σύμφωνα με τις απαιτήσεις της άσκησης, γράφοντας make, τρέχοντας έτσι το Makefile αρχείο που υπάρχει στον φάκελο (το οποίο είναι το ίδιο που έδειξε ο κύριος Ντούλας στην διάλεξη). Το πρόγραμμα κατά την έναρξη περιμένει τα ορίσματα που έχουν περιγραφεί στην εκφώνηση. Αν αυτά δεν δωθούν ή αν είναι λανθασμένα(Πχ αν στο bucket size εισάγουμε έναν αριθμό που δεν είναι ικανός, σαν μέγεθος bytes , να “χωρέσει” τα στοιχεία που περιλαμβάνονται σε ένα bucket) γίνεται αντικατάσταση απο τα default values αυτών των μεταβλητών, που υπάρχουν ορισμένα στο αρχείο definitions.h.

Μόλις ολοκληρωθεί ο παραπάνω έλεγχος, ξεκινά η ανάγνωση του αρχείου και η κατασκευή όλων των απαιτούμενων δομών. Αρχικά δημιουργείται ο ‘κορμός’ των δύο hashtables μέσω της createHashTable. Στην συνέχεια κατασκευάζεται η συνδεδεμένη λίστα μέσω της συνάρτησης InsertList, η οποία λαμβάνει γραμμή γραμμή το κάθε entry, το χωρίζει στα απαιτούμενα πεδία(recordID, patientFirstName, patientLastName, diseaseID, country, entryDate, exitDate) και αν εντωπίσει κάποιο απο αυτά τα πεδία κενά, τότε απορρίπτει το entry, προχωρώντας στο επόμενο. Επίσης ελέγχεται και το πεδίο recordID, ώστε να απορριφθεί η εισαγωγή ενός entry με recordID, το οποίο υπαρχει ήδη στην λίστα, καθώς και τα πεδία entryDate και exitDate, ώστε να διασφαλιστεί το ότι, αν υπάρχει exitDate, είτε θα έπεται, είτε θα είναι ίση με την entryDate (μέσω της checkDate), σε αντίθετη περίπτωση η εγγραφή απορρίπτεται. Κάθε φορά που μία εγγραφή απορρίπτεται, γιατί δεν πληρεί, μία από τις παραπάνω προϋποθέσεις, το πρόγραμμα δεν τερματίζεται, αλλά εμφανίζεται κατάλληλο μήνυμα στον χρήστη και η ανάγνωση συνεχίζει κανονικά με την επόμενη γραμμή του αρχείου. Να σημειωθεί ότι για κάθε ένα από τα strings που εισάγωνται στις δομές, έχει γίνει πρώτα η μετατροπή τους σε κεφαλαία, ώστε να μην υπάρχει θέμα με τις συγκρίσεις και το hash στην συνέχεια.

Κάθε φορά που μία εγγραφή εισάγεται στην λίστα γίνεται και η αντίστοιχη εισαγωγή της στα hashtables. Η δομή του hashtable αποτελείται απο έναν ακέραιο, που κρατά το μέγεθος του συγκεκριμένου hashtable και έναν δείκτη σε δομή HashTableEntry. Κάθε HashTableEntry αποτελείται απο έναν δείκτη σε δομή bucket, που “δείχνει” στο πρώτο bucket,το οποίο δημιουργείται για το συγκεκριμένο entry, καθώς και απο έναν ακέραιο ο οποίος αρχικοποιείται με τον αριθμό των θέσεων που θα έχει κάθε bucket και “κρατά” τον αριθμό των θέσεων, που είναι διαθέσιμες, μόνο στο τελευταίο bucket. Καθώς το πρόγραμμα εκτελείται, καινούρια buckets δημιουργούνται αν, και μόνο αν, ο αριθμός των διαθέσιμων θέσεων του τελευταίου bucket μηδενιστει. Άρα το να κρατάμε τον αριθμό διαθέσιμων θέσεων σε γεμάτα buckets είναι

πλεονασμος. Κάθε bucket αποτελείται απο έναν δείκτη στο επόμενο bucket και απο έναν δείκτη σε BucketEntry. Το BucketEntry, τέλος, αποτελείται απο την την τιμή που αποθηκεύεται στο hashtable καθώς και απο έναν δείκτη στην ρίζα του ζητούμενου BSS tree. Η λογική που ακολουθείται προκειμένου να γίνεται η σωστή δέσμευση του bucket είναι η εξής: αρχικά δεσμεύεται χώρος στην μνήμη που είναι ικανός να χωρέσει τα κ BucketEntry, που υπολογίζεται ,μέσω της συνάρτησης entriesPerBucket, ότι χωράνε σε κάθε bucket συν το μέγεθος του bucket. Μετά, ο pointer, ο οποίος δείχνει στην αρχή του δεσμευμένου bucket, γίνεται cast σε τύπο bucket, και ο χώρος που είναι δεσμευμένος, sizeof(struct Bucket) bytes μετά γίνεται cast σε τύπο BucketEntry, και ανατήθεται στον δείκτη entry του bucket. Η εισαγωγή πραγματοποιείται με την συνάρτηση InsertHashTable, η οποία καλεί με την σειρά της άλλες συναρτήσεις, πχ για να γίνει το hash του στοιχείου, για την κατασκευή του entry, για να γίνει allocate ένα νέο bucket όπου χρειάζεται, κλπ.

Κατά την δημιουργία των HashTables, γίνεται και η δημιουργία και ενημέρωση των AVL trees για κάθε country και disease, κάθε κόμβος των οποίων κρατά την ημερομηνία εισαγωγής των ασθενών, ένα δείκτη στο entry της λίστας του ασθενούς καθώς και μία λίστα, στην οποία, αποθηκεύονται κόμβοι δέντρου πού έχουν την ίδια ημερομηνία, ώστε να διευκολύνεται η αναζήτηση στο δέντρο. Αρχικά καλείται η συνάρτηση InsertTreeNode. Ξεκινώντας, η ημερομηνία, που μέχρι τότε είχε την μορφή string, μετατρέπεται μέσω της CreateDate, σε struct Date. Συνεχίζοντας η αναδρομική συνάρτηση διασχίζει το δέντρο, συγκρίνοντας την ημερομηνία, του προς εισαγωγή κόμβου, με την ημερομηνία του root των υποδέντρων, με την συνάρτηση (checkEntryDate) , και ανάλογα η διάσχιση συνεχίζεται στο αριστερο υποδέντρο(αν είναι μεγαλύτερη), στο δεξί (αν είναι μικρότερη), ενώ αν οι δύο ημερομηνίες είναι ίσες, τότε ο κόμβος εισάγεται στην treeList λίστα(που υπάρχει σε κάθε κόμβο του δέντρου. Αντί να συνεχίζουμε αριστερά ή δεξιά του κόμβου πηγαίνουμε κατά έναν τρόπο ευθεία). Στην συνέχια, ελέγχεται αν το δέντρο συνεχίζει να είναι balanced, δηλαδή, αν η διαφορά του ύψους κάθε υποδέντρου, είναι το πολύ ίσο με 1 ή -1. Αν αυτό δεν ισχύει, τότε υπάρχουν 4 υποπεριπτώσεις που ελέγχονται και επανέρχεται και πα΄λι η ισορροπία στο δέντρο, μέσω των συναρτήσεων RotateRight, RotateLeft.

Μόλις γίνει η δημιουργία όλων των απαιτούμενων δομών, δίνεται η δυνατότητα στον χρήστη να υποβάλει τα ερωτήματά του, μέσω των συναρτήσεων που περιγράφονται στην εκφώνηση. Να διευκρινιστεί εδώ, ότι κάθε τιμή πριν εισαχθεί, ή πριν γίνει αναζήτησή με βάση αυτήν σε κάποια δομή μετατρέπεται σε κεφαλαία μέσω της ConvertToUpper, ώστε να γίνει η διαδικασία case insensitive. Επίσης, και η ανάγνωση των εντολών που δίνονται από τον χρήστη θα είναι case insensitive, δηλαδή η /InsertPatientRecord, η /INSERTPATIENTRECORD και η /InsertPAtientRECord θεωρούνται το ίδιο.

- /InsertPatientRecord, η λειτουργία αυτή δρά κατά παρόμοιο τρόπο, με αυτόν που περιγράφτηκε παραπάνω. Γίνεται η εισαγωγή της νέας εγγραφής, με ενημέρωση όλων των δομών ή αν απορρίπτεται αν δεν είναι σωστή.
- /recordPatientExit, η οποία ενημερώνει την εγγραφή με το κατάλληλο recordID, τροποποιώντας την ημερομηνία εξοδου του ασθενή, μέσω της InsertExitDate. Αν δεν υπάρχει ημερομηνία εξόδου τότε προστήθεται, ενώ αν υπάρχει, τότε ενημερώνεται, με μόνο περιορισμό και στις δύο περιπτώσεις, η ημερομηνία εξόδου να είναι μεγαλύτερη ή το πολύ ίση με την ημερομηνία εισαγωγής. Αν η τροποποίηση πραγματοποιηθεί, τότε η τροποποιημένη εγγραφή εμφανίζεται στον χρήστη, αλλιώς εμφανίζεται μήνυμα λάθους, που εξηγεί τους λόγους, μη πραγματοποίησης της.
- /globalDiseaseStats, η οποία εμφανίζει (για το δωσμένο διάστημα, αν δωθεί) πόσα κρούσματα, έχουν σημειωθεί για τον κάθε ιό, μέσω της συνάρτησης DiseaseStatistics. Αν δεν δωθεί ζεύγος ημερομηνιών, τότε προκειμένου να μην χρειαστεί διαφορετική υλοποίηση, ορίζεται αυθέρετα ένα μεγάλο διάστημα (1-1-1500 με 30-12-2030). Έτσι διασφαλίζεται ότι θα γίνει καταμέτρηση, όλων των καταγεγραμένων περιστατικών. Η παραπάνω συνάρτηση για κάθε ασθένεια που υπάρχει στο dHashTable, καλεί την συνάρτηση GetCases, η οποία διατρέχει το δέντρο που κρατά τις ημερομηνίες εισαγωγής, για όλα τα κρούσματα της συγκεκριμένης ίωσης και επιστρέφει τον αριθμό αυτών που σημειώθηκαν στο δωσμένο διάστημα ή όλα το σύνολο όλων των κόμβων για το αυθαίρετο διάστημα. Κατόπιν τυπώνονται τα αποτελέσματα στον χρήστη.
- /diseaseFrequency, η οποία τυπώνει τον αριθμό των κρουσμάτων για την συγκεκριμένη ασθένεια (και την συγκεκριμένη χώρα, αν δωθεί ώς όρισμα) στο συγκεκριμένο χρονικό πλαίσιο. Για τον λόγο αυτόν καλείται, όπως και πριν η DiseaseStatistics και μετέπειτα η GetCases, με την μόνη διαφορά ότι πλέον δεν ψάχνουμε κάθε ασθένεια, αλλά μετράμε τα cases για την συγκεκριμένη

ασθένεια(και για την συγκεκριμένη χώρα, αν υπάρχει, πηγαίνοντας στην θέση της λίστας που είναι η συγκεκριμένη εγγραφη και συγκρίνεται η χώρα).

- /numCurrentPatients, η οποία εμφανίζει πόσοι ασθενείες νοσηλεύονται ακόμα (δηλαδή σε πόσες εγγραφές δεν υπάρχει ημερομηνία εξόδου) είτε για όλες τις ασθένειες, είτε για μια συγκεκριμένη (αν δίνεται όρισμα), μέ την χρήση της συνάρτησης CurrentPatients. όταν δεν δίνεται όρισμα, τότε διατρέχεται το disease Hashtable και για κάθε ασθένεια καλείται η συνάρτηση NoExitDate, η οπόια διατρέχει αναδρομικά όλο το δέντρο που κρατά τις ημερομημίες για την συγκεκριμένη ασθένεια. Για κάθε κόμβο στο δέντρο ελέγχεται η εγγραφή του στην λίστα, μέσω του δείκτη και αν το exitDate="-" αυξάνεται ο μετρητής. Αν δωθεί συγκεκριμένη ασθένεια, τότε κοιτάμε στο συγκεκριμένο entry του hashtable, αν υπάρχει. Αν υπάρχει τότε καλείται η NoExitDate, μόνο για αυτή την ασθένεια, αλλίως τυπώνεται μήνυμα λάθους (στα πλαίσια του script εξέτασης τυπώνεται η ασθένεια και 0).
- /topk-Countries και /topk-diseases . Οι δύο αυτές εντολές έχουν αντίστοιχη λειτουργικότητα και εξυπηρετούνται απο τις ίδιες συναρτήσεις Topk και Cases καθώς και τις συναρτήσεις που χρειάζονται για την κατασκευή, την διατήρηση και την διαγραφή του maxheap με χρήση binary tree, που απαιτείται σε αυτά τα ερωτήματα. Θα αναλυθεί η λογική που ακολουθείται στην /topk-Countries, για την οποία τυπώνονται οι k πρώτες χώρες με τα περισσότερα κρούσματα για την ασθένεια που δίνεται σαν όρισμα, στο χρονικό διάστημα που δίνεται, αν δίνεται. Αρχικά καλείται η συνάρτηση Topk. Αν δεν δίνονται ημερομηνίες, τότε αυτές τήθενται σε ένα αυθαίρετα μεγάλο διάστημα(1-1-1500 με 30-12-2030). Για κάθε χώρα που υπάρχει στο hashtable με τις χώρες, καλείται η συνάρτηση Cases, η οποία κάνει αναζήτηση, στο δέντρο της κάθε χώρας, και επιστρέφει το σύνολο των εγγραφών, που βρίσκονται στην επιθυμητή χώρα. Κατόπιν η τιμή αυτή εισάγεται στο maxheap, το οποίο δημιουργείται εκείνη την στιγμή, μέσω της InsertHeap. Το maxheap, υλοποιείται σύμφωνα με τις απαιτήσεις την άσκησης, με την μορφή binary tree. Η εισαγωγή στο maxheap, γίνεται με βάση το ελάχιστο ύψος, του κάθε υποδέντρου. Το heap, είναι ένα δέντρο, του οποίου τα φύλλα βρίσκονται, είτε στο τελευταίο, είτε στο πρότελευταίο επίπεδο και η εισαγωγή ενός νέου κόμβου, γίνεται όσο το δυνατόν πιο αριστερά, με μόνη προυπόθεση να πληρείται η παραπάνω ιδιότητα. Έτσι, η βασική λογική πίσω από τον τρόπο με τον οποίο γίνεται η εισαγωγή στο πρόγραμμα μου, είναι η αναδρομική διάσχιση του δέντρου, με το νέο υποδέντρο που επιλέγεται να έχει μικρότερο ελάχιστο βάθος από το δεύτερο υποδέντρο, παιδί του εκάστοτε root. Όταν η συνάρτηση πλέον φτάσει σε NULL, τότε γίνεται εκεί η δημιουργία του κόμβου. Παράλληλα, αναδρομικά ελέγχεται, και αν κάποιος κόμβος-παιδί έχει μεγαλύτερη τιμή από αυτή του κόμβου πατέρα. Αν αυτό ισχύει, τότε καλείται η SwapHeapNodes, η οποία αλλάζει τις τιμές των δύο κόμβων, ώστε να διασφαλιστεί η συνθήκη του maxheap, δηλαδή οτι κάθε κόμβος-πατέρας, έχει τιμή μεγαλύτερη (ή στην περίπτωση μας και ίση) απο κάθε ένα απο τους κόμβους παιδία του. Όταν η παραπάνω διαδικασία ολοκληρωθεί για όλες τις χώρες, θα έχει δημιουργηθεί και ο maxheap, και στην ρίζα του θα βρίσκεται η χώρα με τα περισσότερα κρούσματα (στο δωσμένο χρονικό διάστημα) για την συγκεκριμένη ασθένεια. Για να τυπωθούν οι k πρώτες απο αυτές καλείται η συνάρτηση PopHeapNode, η οποία αφαιρεί κάθε φορά την τιμή της ρίζας, στην θέση της βάζει την τιμή του φύλλου (το struct του αποδεσμεύεται) που βρίσκεται πιο δεξιά στο τελευταίο επίπεδο του maxheap και καλεί την Heapify, προκειμένου να ανέβει και πα΄λι στην ρίζα το μεγαλύτερο στοιχείο του heap. Η Heapify, είναι μια ανδρομική συνάρτηση που διασχίζει το δέντρο-heap και διορθώνει ανωμαλίες, ώστε το μέγιστο στοιχείο να βρεθεί στην ρίζα. Άρα κάθε φορά που καλείται η PopHeapNode, έχουμε το μέγιστο στοιχείο στα χέρια μας και το επόμενο μέγιστο στην ρίζα του heap. Στην περίπτωση που δύο ή παραπάνω χώρες έχουν τον ίδιο αριθμό cases, τότε τυπώνονται όλες αυτές οι χώρες και μοιράζονται την ίδια θέση. Πχ, αν το κ=2 και η μέγιστη τιμή του heap είναι 20, αλλά υπάρχει και δεύτερη χώρα που έχει 20 cases, τότε θα τυπωθούν και οι δύο χώρες ώς πρώτες και μετά θα τυπωθεί και η αμέσως επόμενη χώρα, ώς δεύτερη (συνολικά τυπώνονται 3 χώρες). Μόλις τυπωθούν οι κ χώρες, συνεχίζεται να καλείται η PopHeapNode, μέχρι να αποδσμευτούν όλα τα nodes του heap. Αν η εντολή /topk-Countries κληθεί ξανά, τότε το heap, δημιουργείται εκ νέου.
- /exit, γίνονται οι απαραίτητες αποδεσμεύσεις μνήμης και το πρόγραμμα τερματίζεται.
- /PrintList, τυπώνει τα περιεχόμενα της λίστας
- /PrintHashTables, τυπώνει το περιεχόμενο, των δύο hashtables.

Κατά την εκτέλεση του προγράμματος με το δεύτερο script, το οποίο ανέβηκε στο piazza του μαθήματος, σε όλα τα .cmd αρχεία, παίρνω pass, εκτός απου το 3ο, στο οποίο "κολλάει" και μετά απο λίγη ώρα παίρνω fail. Επίσης, αν διαγράψω την εντολή που υπάρχει στο αρχείο αυτό και την ξαναγράψω ή πατήσω enter, ώστε να μπεί ο χαρακτήρας αλλαγής γραμμής, παίρνω pass και για

αυτό το .cmd αρχείο. Έκανα σχετική ερώτηση στο piazza και ο instructor George Panagiotopoulos μου απάντησε ότι είμαι οκ και απλά να κάνω μια αναφορά στο README μου.