ΣΥΣΠΡΟ Project1

sdi1400195 Σοφία Στυλιανού Νικολαΐδου.
Compile : make

Το πρόγραμμα αρχικά διαβαζει το balances file και αποθηκεύει τα περιεχόμενα του στις δομες. Αν βρει λάθος εγγραφη κατα διάρκεια διαβάσματος, τερματίζει.
Στη συνέχεια διαβάζει το transactions file αποθηκεύει και εκτελεί τις συναλλαγές και σε περίπτωση λανθασμένης συναλλαγής εμφανίζει μήνυμα και την απορρίπτει.
Έπειτα περιμένει εντολές απο τον χρήστη.

Balances.c:

load_balances: ανοίγει το αρχείο διαβάζει τις εγγραφές και δημιουργεί και ενημερώνει εναν πίνακα wallet* wallet_arr.
Η δομή **wallet**: αποθηκεύει τις πληροφορίες που έχει ο κάθε χρήστης στο πορτοφόλι του. Ενα string walletId μοναδικο για κάθε χρήστη, τον αριθμό των bitcoins που κατέχει, το συνολικό ποσό που έχει στο πορτοφόλι του και μια λίστα με τα bitcoins του.

Συγκεκριμένα η λίστα αυτή, **bitcoin_list** αποτελείται απο κόμβους που περιέχουν την δομή bitcoin. Επειδή η δομή αυτή μεταβάλλεται μετα τις συναλλαγες, περα απο το bitcoin έχει και εναν int value, που καθορίζει τι ποσό κατέχει ο χρήστης απο το bitcoin.
Κάθε δομή **bitcoin** έχει ενα μοναδικό int id, τον αριθμο των συναλλαγών που είχε συμμετοχή, και ένα δείκτη στην ρίζα του δέντρου του tree_node* tree.

Επίσης, δημιουργείται ένα **hashTable για τα bitcoins bitC_hashT** με μέγεθος που προκυπτει πολλαπλασιάζοντας τον defined μέσο αριθμό των bitcoins του χρήστη με τον αριθμό των χρηστών που έχουν διαβαστεί απο το balances file και πολλαπλασιαζοντας με έναν παράγοντα 1.3 ωστε να δημιουργηθεί load factor περίπου 70%, τέλος για τον καλυτερο κατακερματισμο, επιλέγεται ο επόμενος πρώτος αυτού. Σε αυτό το hash table αποθηκεύονται bitcoin * για την γρηγορη αναζήτηση των bitcoins με κριτήριο το bitcoinId.

Δημιουργουνται δύο **hash tables sr_hashT** που αποτελούν τον κατακερματισμο των senders και receivers. Κάθε bucket, έχει για όσες εγγραφες χωράνε απο το -b, έναν pointer σε wallet και έναν pointer σε **transact_list**, ο οποιος δείχνει σε μια λίστα που αποθηκεύονται τα transactions του χρήστη ως sender στο sender sr_hashT και ως receiver στο receiver sr_hashT.
Ο κατακερματισμός στους πίνακες γίνεται με βάση το όνομα-walletId. Προστίθενται οι τιμές των χαρακτήρων και επιστρέφεται το υπόλοιπο της διαίρεσης με το μέγεθος του hash table.

Έτσι για κάθε γραμμή του αρχείου καλέιται η insert_newWallet και αποθηκεύεται το wallet στον πίνακα wallet_arr. Αν αποτύχει επιστρέφει και τερματίζει το πρόγραμμα.
Τέλος αποθηκεύονται οι pointers του wallet στους πίνακες κατακερματισμού sender_ht και receiver_ht και αντίστοιχα τα bitcoins στο bitcoin hash table (bitC_hashT).

insert_newWallet: διαβάζει μια γραμμή αρχείου balances που δίνεται, κάνει τους κατάλληλους ελέγχους για διπλότυπο bitcoinId ή walletId και δημιουργεί το wallet.

addToHash_bitC: προσθέτει ενα bitcoin στον πίνακα κατακερματισμού bitC_hashT.
add_bitCoin: προστίθεται ενα bitcoin στην λίστα του wallet με τα bitcoins.
create_hashT: δεσμεύεται και αρχικοποιείται το hash table για τους senders και receiivers.
addToHash_walletId: προστίθεται ένα wallet στην κατάλληλη θέση των senders και receivers.

Οι υπόλοιπες συναρτήσεις του αρχείου, είναι για εκτύπωση, αναζήτηση και αποδέσμευση μνημης.

Transactions.c

load_transactions: Αντίστοιχα παίρνει ως όρισμα το αρχείο transactions ,ενημερώνει τις δομές και εκτελεί τις συναλλαγες.

Δημιουργείται ένα **hash table tr_hashT** για την αποθήκευση των συναλλαγων με βάση το Id τους. Πάλι το μέγεθος είναι defined ως 255. Κάθε bucket έχει defined TR_BUCK εγγραφές απο δομές transaction. Τα transactions κατακερματίζονται με βάση το transactionId όπως τα walletIds.

Η δομή **transaction** αποθηκεύει της πληροφορίες μίας συναλλαγης. Έχει ενα string descript, που είναι η περιγραφη της συναλλαγης, εναν char* id, ενα struct tm για την αποθήκευση της ημερομηνίας και εναν int amount για το ποσο.

Στην αρχή αρχικοποιεί το tr_hashT και για κάθε γραμμή του αρχείου που διαβάζει καλεί την insert_newTransaction.

insert_newTransaction: Διαβάζει την γραμμή του αρχείου , δημιουργεί ενα transaction, το αποθηκεύει στο hash_table και ενημερώνει τις λίστες του sender και receiver με τον pointer του transaction.
Τέλος το εκτελεί και ενημερώνει τα δέντρα των bitcoins που συμμετείχαν.
Προκειμένου ένα transaction να γίνει αποδεκτό, ελέγχεται αν υπάρχει ξανά το id, αν ο sender και ο receiver έχουν αποθηκευμένα wallets, αν είναι αρκετό το συνολικό ποσό στο wallet του sender για την εκτέλεση της συναλλαγης και τέλος η εγκυρότητα ημερομηνία.

addToHash_tr: Δημιουργείται και αποθηκεύεται το transaction στο hash table tr_hashT.
add_trasnact: Προστίθεται στην λίστα με τα transactions του sender και του receiver ενας κόμβος με τον pointer του transaction.
swap_listNode,receive_bitc: για την εκτέλεση της συναλλαγης, ανάλογα την περίπτωση ενημερώνονται οι λίστες με τα bitcoins του sender ,receiver. Είτε προστίθεται ενας κόμβος με το καινουργιο bitcoin στον παραληπτη, είτε ενημερώνεται το ποσό του ήδη υπάρχοντος bitcoin στον παραλήπτη, είτε αποκτά εξ ολοκλήρου τον κόμβο με το bitcoin και ο αποστολέας, δεν έχει πλέον αυτό το bitcoin.

update_tree: δημιουργείται και ενημερώνεται αναδρομικά το δέντρο του bitcoin. Πρώτα ελέγχονται τα δεξιά παιδιά και μόλις βρεθεί θέση φύλλο με το walletId του sender γίνεται εισαγωγη.

δομή tree_node: υπάρχει ένα δεξιό και ένα αριστερό παιδί, ενας pointer σε transaction και ένας pointer σε wallet, καθώς και ένας int που υποδεικνύει σε κάθε φάση το ποσό που έχει ο receiver και ο sender απο το bitcoin.
Οι κόμβοι πατέρες είναι εκείνοι που έχουν ποιντερ σε συναλλαγή και pointer στον sender ,ενώ τα φύλλα έχουν μόνο pointer σε wallet (sender-δεξιο παιδί, receiver-αριστερό παιδί) και το ποσό.

exec_transaction: Διατρέχει την λίστα με τα bitcoins του sender, επιλέγει το πρώτο διαθέσιμο και εκτελέι την συναλλαγή, αν δεν φτάνει να καλυφθεί απο το πρώτο στην λίστα προχωράει στο επόμενο κοκ. Αν ο sender δώσει όλο το ποσό ενός bitcoin διαγράφεται απο την λίστα του ως sender και παραμένει στους παραλήπτες.

Commands.c:
Περιλαμβάνει όλες τις εντολές που δίνει ο χρήστης απο το command line.
requestTransaction: Παρόμοια με την insert_newTransaction, διαφέρει στον χειρισμό του id και ρης ημερομηνίας. Το id δημιουργείται παίρνοντας την max τιμή απο προηγούμενα transactionsIds και αυξάνοντας την κατα 1. Σχετικά με την ημερομηνία αν δεν δοθέι, αποθηκεύεται η τρέχουσα.

requestTransactions: Μαζί με την εντολή δίνεται ένα flag για το αν η γραμμή περιλαμβάνει ερωτηματικο. Flag=0 σημαίνει οτι δεν περιλαμβάνει " ; " και αναφέρεται σε αρχείο. Οπότε ανοίγει το αρχείο και για κάθε γραμμη καλεί την requestTransaction, διαφορετικά, flag=1 καλείται απευθείας η requestTransaction για την δοθείσα γραμμή. Η εντολή αναμένει οτι μετά το ερωτηματικό υπαχει αλλαγή γραμμής γιατι το διάβασμα γίνεται με την χρήση της getline.

findEarn_Paym: Καλείται για την findEarnings και findPayments, ο διαχωρισμος γίνεται με flag. Μετά την διασφάλιση εγκυρότητας του wallet, αποφασίζει αν υπάρχει χρονικός περιορισμός. Ανάλογα την περίπτωση καλείται η **totalEarn_Paym** η υπεύθυνη για το αθροισμα του ζητούμενου ποσού εντός χρονικών περιορισμών και την εκτύπωση των transactions.

walletStatus: εκτυπώνει τις ζητούμενες πληροφορίες απο το wallet.

bitcoinStatus: Εκτυπώνει το συνολικό αριθμό transactions που συμμετείχε το bitcoin και βρίσκει το δεξιότερο φύλλο του δέντρου για το ποσό που δεν έχει χρησιμοποιηθεί σε καμία συναλλαγη.

traceCoin: εκτυπώνει με την βοήθεια της **TransactsOfbitc** τις συναλλαγες που έχει συμμετάσχει το bitcoin. Ουσιαστικά διατρέχει το δέντρο και εκτυπώνει τους κόμβους πατέρες (transactions).