ΑΜ : 1115201400024
Δημήτριος Γάγγας

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ

Εργασία 1

Υλοποιήθηκε το project επιτυχώς
σε C++11 με λειτουργικό Linux χωρίς τη χρήση STL ή άλλων βιλιοθηκών.

- **[Optimality Note]** Η δημιουργία διπλότυπης πληροφορίας έχει αποφευχθεί.
- Ελέγχθηκαν επιτυχώς με Valgrind .Συνεπώς δεν υπάρχουν memory leaks.
- Υλοποιήθηκαν όλοι οι απαιτούμενοι έλεγχοι.
- Χρησιμοποιήθηκε το git και το gitkraken για οπτικοποίηση των εκάστοτε αλλαγών αλλά και για την ευκολία επιστροφής σε προηγούμενα στάδια ανάπτυξης του κώδικα.
- Δημιουργήθηκαν template δομές για linked list και HashTable .
- Περιέχεται makefile που μεταγλωττίζεται με make.
- Έχουν υλοποιηθεί όλες οι απαιτήσεις της εκφώνησης για τα ορίσματα με όποια σειρά και να δοθούν τα flags καθώς και οι απαραίτητοι έλεγχοι.

  $../bitcoin -a bitCoinBalancesFile –t transactionsFile -v bitCoinValue –h1 senderHashtableNumOfEntries –h2 receiverHashtableNumOfEntries –b bucketSize

- Επίσης, αν δε δωθούν κάποια flags τοτε το προγραμμα ζητάει την ενημέρωση τους.
- Περιέχεται και ενα εκτελέσιμο μέσα στο φάκελο με ονομασια **bitcoin**.

Αρχεία .cpp/.tpp/.h που δημιουργήθηκαν :

• APIfunctions.cpp	APIfunctions.h
• assistantFunctions.cpp	assistantFunctions.h
• hashFunction.cpp	hashFunction.h
• synchroniseFunctions.cpp	synchroniseFunctions.h
• transacHashMap.cpp	transacHashMap.h
• bitcoin.cpp	bitcoin.h
• Transaction.cpp	transaction.h
• bitcoinTree.cpp	bitcoinTree.h
• myBucket.tpp	myBucket.h
• Wallet.cpp	wallet.h
• date.cpp	date.h
• myString.cpp	myString.h
• myHashMap.h	
• mylinkedList.h	
• ErrorsCodes.h	

Βασικές Δομές που υλοποιήθηκαν

- Δομή linked list με templates + iterator για να την διασχίζει.

- Δομή Tree για να αποθηκεύει το ιστορικό συναλλαγών για κάθε btc.

- Δομή transacBucket που γινεται new με το BucketSize που ζητειται απο την εκφώνηση Και αποθηκεύει το μέγιστο δυνατό αριθμό εγγραφών που μπορεί να χωρέσει.Οι εγγραφές περιέχουν το sender/receiver walletId + ενα object linked list Που περιέχει pointer σε transactions (Αναλύεται παρακάτω).

- Δομή myBucket που απλα είναι περιέχει τα data //Πρακτικά δε χρειάζεται καθώς θα μπορούσα απλα να αποθήκευα το linked list<T> στο HashMap, Αλλα στη προηγούμενη υλοποίηση ειχα μια δομη HashMap που τις πέρναγα τι τυπου bucket θέλω [transacBucket ή myBucket] και χρειαζομουν συμμετρικότητα στα layers.
  Π.χ myHashMap<myBucket_chain <wallet> > ή
  myHashMap<recordsBucket_chain <bitcoin> >

- Δομή myHashMap που είναι ένα απλό template HashTable χώρις τη χρήση των recordsBuckets.Δηλαδη στα collisions απλά περιέχει μια συνδεδεμένη λίστα με τα αντικείμενα (aka myBucket in my implementation).

- Δομή transacHashMap είναι αντίστοιχα ένα template HashTable που περιέχει Και το layer με τα buckets και τα records που ζητείται στην εκφώνηση για hashTables

των sender και receiver.

Παραπάνω υλοποιήσεις worth mentioning

- Class myString που έχει σε έναν πολύ μικρό βαθμό τις δυνατότητες της δομης std::string της STL.

- Class date για τους οποίους κάνω overload του operators συγκρισης.

- Class sync ,η οποία είναι υπεύθυνη για το insertion των συναλλαγών Και για το συγχρονισμό όλων των δομών.