Пояснювальна записка
Індивідуального завдання
На тему “Закрите хешування(відкрита адресація)”
Акав Нікіта КМ-31

Мета роботи: використовуючи метод закритого хешування, розробити макет роботи із логіном та паролем користувача, для доступу до якоїсь інформації.

Опис алгоритму: тут буде пояснення роботи кожної з функцій, які використовуються в програмі. Функції представлені не у порядку виконання роботи, а зверху вниз.

Назва функції	Що вона робить
Hash_password	Ця функція приймає пароль користувача і перероблює у число, яке не буде перевищувати 30ти, і буде використано в якості індексу комірки. Вона це робить таким чином, що кожен символ паролю перетворюється на числове значення з unicode.
Hash_password_in_table	Ця функція приймає пароль і логін користувача, з’єднує їх в один рядок, потім вже з бібліотеки hashlib вона бере метод sha256, аби закодувати пароль і логін, які будуть вставлені в таблицю як 16ти символьний рядок для отримання доступу.
insert	Ця функція створює аккаунт користувача. Вона спочатку приймає логін, потім перевіряє його на унікальність за допомогою наступної функції. Потім приймає пароль та інформацію по акаунту і далі, використовуючи попередні функції й функцію “Insert_in_table ” вставляє данні в хеш-таблицю.
Check_login	Ця функція переходить до файлу hash.txt і перевіряє кожен рядок.

	Якщо рядок пустий або там [DELETED], то функція його пропускає, якщо ж ні, то функція розуміє, що розбитий рядок символами “:”, переходить до значення, де повинен стояти логін і співставляє його з тим, що користувач написав зараз.
Insert_in_table	Ця функція вже вписує в таблицю аккаунт, якщо строка пуста або [DELETED]. Також у разі, якщо недостатньо рядків, то вона просто їх створює і вписує в файл хешоване значення паролю і логіну в 16 символів, логін, а також інформація.
Access_to_account	Ця функція теж приймає логін та пароль, перетворює їх на хешоване значення і пароль на індекс, знаходить потрібний рядок, співставляє хешоване значення в таблиці із тим, що отримала і вже або надає доступ, або ні.
Delete_account	Ця функція, приймаючи пароль та логін, робить усе те саме, що і попередня, але після того, як знаходить потрібний аккаунт, видаляє його зі списку і залишає в рядку [DELETED]. Це зроблено для того, аби у випадку, якщо цей рядок став частиною колізії, функція access_to_account розуміла, що потрібно пройти далі, а не припинити на цьому місці пошук.

Опис алгоритму роботи із програмою і самої програми: на початку програми користувачу дається вибір серед 4-ох опцій:

I. Create account: після вибору, програма запитує користувача про логін, пароль та інформацію про аккаунт, перевіряє логін на унікальність, перетворює пароль на індекс комірки для таблиці і логін з паролем на хешоване значення, яке буде вписане в таблицю. Вже після всього цього вставляє інформацію в таблицю.

II. Get access to account: після вибору, програма запитує користувача про логін та пароль, після чого пароль конвертується у індекс і програма одразу переходить на нього у таблиці, потім логін з паролем перетворюються на їх хешоване значення і співставляється з тим, що знаходиться у таблиці. Якщо це теж саме значення- виводиться інформація, якщо ні, то користувачу виводиться помилка.

III. Delete account: після вибору, програма запитує користувача про логін та пароль, після чого пароль конвертується у індекс і програма одразу переходить на нього у таблиці, потім логін з паролем перетворюються на їх хешоване значення і співставляється з тим, що знаходиться у таблиці. Якщо це теж саме значення- програма видаляє акаунт і заміняє його [DELETED], якщо не, то виводиться помилка.

IV. Exit: програма завершує роботу.

Всього може бути 30 заповнених комірок, бо індекси ми саме таким чином присвоюємо. У випадку колізії на 30тій комірці, новий акаунт переходить на 1шу і шукає з самого початку таблиці.

Загальна інформація по хешуванню:

Хешування - це перетворення вхідного масиву даних певного типу і довільної довжини у вихідний бітовий рядок фіксованої довжин

Закрите хешування або Метод відкритої адресації - це технологія вирішення колізій, що припускає зберігання записів у самій хеш-таблиці.

Загальна складність алгоритму $\theta(1)$, бо пошук йде по індексу, і у випадку відсутності колізії, пошук починається і завершується на потрібному нам місці. У найгіршому випадку складність буде $\theta(к)$, де к- загальна кількість рядків, в нашому випадку 30. Це означає, що таблиця повністю заповнена, через що програмі доведеться перепройти кожнен з рядків, дійти до того, з якого починалося і, не знайшовши вільну комірку, вивести помилку.

Колізія- ситуація, коли хешоване значення ключів співпадають, а ключі різні. Якщо ж це стосується індексів, то ми використовуємо метод відкритої адресації, тобто змінюємо індекс так, аби найефективніше знайти пусту комірку і вписати туди потрібну

інформацію. У випадку, якщо це стосується хешованого значення, яке записане в таблиці, з яким ми порівнюємо при пошуку, то це може виявитися великою проблемою, бо знайшовши подібну комбінацію зловмисник зможе отримати доступ до інформації.

Найголовніша задача хеш-функції, якнайефективніше порахувати хеш-значення й звести до мінімума випадки колізії, а також при зливі данних захистити їх, бо у хеш-таблиці вони будуть знаходитися у вигляді вже хеш-значення.