**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Звіт**

**Перша задача, алгоритми та складність**

**Предметна область : “Продукти”**

**Підготував студент**

**2 курсу**

**Групи К29**

**Опанюк Микита**

Опис предметної області :

«Вариант 7   
Предметная область  Продуктовый магазин   
Объекты  Категория продукта, Продукт   
Примечание  Продукты  в  магазине   
сгруппированы  по  категориям.   
Для  каждой  категории   
определено  множество   
продуктов.»

**Інтерфейс :** консоль.

В задачці побудовано 3 класи : class products{}; саме в цьому класі йде основна реалізація предметної області. А саме :

Клас працює з 2 типами структур :

struct type {

string group; - назва різновиду.

unsigned int hash\_key; - відповідно ключ, який підраховується ф-ю hash\_string.

vector<product\_element \*> array\_of\_products; - вектор вказівників на продукти вказаного типу.

}; - використовується для збереження різновидів продуктів.

struct product\_element {

string group; - назва різновиду продуктів, до якого відноситься вказаний.

string name; - відповідно ім’я.

double price; - ціна (для інформативності).

unsigned int hash\_key; - відповідно ключ, який підраховується ф-ю hash\_string.

};

vector<type \*> type\_of\_products; - масив типів продуктів в магазині, де кожен елемент масиву має вказівник на масив самих продуктів.

int check\_is\_product(int pos, string product); - перевіряє наявність вказаного продукту в магазині, де pos – це індекс елемента масива типів продуктів.

int check\_is\_type(string product); - перевіряє наявність відповідно типу продуктів в магазині.

void new\_data(); - додавання нового продукту з перевіркою на наявність типу цього продукту (якщо тип відсутній, то створюється новий), відповідно присутня перевірка того чи існує вже вказаний продукт.

void delete\_data(product\_element \*name); - видаляє з каталогу вказаний продукт (зменшує масив продуктів у відповідного типу)

void save\_changing(); - зберігає всі зміни каталогу, перезаписуючи файл, на якому зберігаються дані що до продуктів.

void read\_data\_from\_file(); - зчитує данні що до продуктів з файлу.

unsigned int hash\_string(string key); - хешує імена груп/продуктів для подальшої роботи з хеш-таблицею (в подальшому дане значення використовуватиметься як «ключ»).

2 інші класи відповідно використовуються в ідеальному хешуванні.

**“Ідеальне хешування”**

За основу написання даних класів взята інформація про Ідеальне хешування за Корменом “Алгоритми. Побудова та Аналіз”.

**Опис :**

1) хешуються назви продуктів : (використання прикладу універсальної функциії хешування з кормена), а саме відбувається переведення string в int з використання ASCII коду кожної букви для створення першого хеш-ключа для першого етапу.

2) Настає перший етап ідеального хешування. Ініціалізуємо основну хеш-таблицю :

б) Порахували перші хеш-ключі, за допомогою яких зберігатимуться значення у вигляді пари “ключ-значення” в vector<vector<int> > elements\_in\_cells.

3) Після того, як для всіх значень підраховані хеш-ключі, переходимо до другого етапу:

а) Другий етап хешування для того, щоб усунути колізії (коли одному хеш-ключу відповідають декілька продуктів), що виникли під час хешування першого етапу. Для цього ініціалізуємо додаткові міні-таблиці для кожного відповідного хеш-ключа, аналогічно ініціалізуємо відповідно кожному ключу власні зміні для використання хеш-функції.

а) Для таблиці зберігаються змінні, потрібні для універсальної фунції хешування (в данному випадку тепер ми працюємо не з строками, а з отриманими хеш ключами), змінні напряму залежать від ключів, а саме знаходимо найбільше серед них значення, відносного нього знаходимо наступне просте число за допомогою використання функції successor\_pime. Коли знаходимо всі потрібні нам змінні, переходимо до повторного хешування продуктів, але вже за допомогою нової функції.

б) Але є одна проблемма – існує ймовірність виникнення колізій і 2 етапу, тому я роблю перевірку, і я в результаті існує колізія, то я повторно хешую дану міні-таблицю з новими зміними для хеш-функції. Це повторюється до тих, поки не буде колізії. На цьому моменті програма завершає основну роботу.

4) Пошук елементів відбувається таким чином : задаємо елемент, спочатку рахуємо на 1 етапі хеш-ключ, порівнюючи з іншими ключами, переходимо до міні-таблиці, для відповідної хеш-функції рахуємо хеш-ключ, і як результат знаходимо елемент.

**Приклад : Внесено до файлу 4 продукти : ковбаса, шоколад, цукерка «Ромашка», курятина.**

**Відповідно при внесенні цих продуктів було вказано групи, і як результат : отримали 2 групи :**

1. **М’ясні продукти : ковбаса, курятина.**
2. **Солодощі : шоколад, «Ромашка».**

**При додаванні цих продуктів до бази даних «магазин» кожен продукт отримує свій ключ, що підраховується за допомогою функції hash\_string();**

**Завершується обробка файлу, переходимо до хешування:**

**Спочатку хешуються типи продуктів : відповідно так як всього 2 типи, то хеш-ключі вони можуть отримати 0 або 1. Далі переходимо до 2 етапу хешування для отримання хеш-таблиці з кращими результатами хеш-ключів (0, 1, 2 або 3).**

**Після запиту до користувача на перевірку групи отримуємо результат, чи є вказаним ним група в базі даних, чи нема. Якщо ж є, то для відповідної групи проводимо хешування ключів продуктів, отримуючи хеш ключі. І робимо відповідний запит на перевірку.**

**Плюси:** Дане хешування дозволяє нам практично уникнути колізій завдяки використанню 2 етапу хешування, пошук значення (різних типів) відбувається за O(n).