

Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інформаційних систем

Алгоритми та складність

Лабораторний проект №1 «Ідеальне хешування»

Звіт

*Виконав:*

*студент групи К-29*

***Кожановський Олександр Сергійович***

Київ-2018

1. **Умова завдання.**

Реалізувати ідеальне хешування

1. **Основні поняття**

*Хеш-функція* — функція, що перетворює вхідні дані довільного розміру в дані фіксованого розміру.

*Хешування* — перетворення масиву вхідних даних довільної довжини в бітовий рядок фіксованої довжини за певним алгоритмом.

*Хеш-таблиця* — структура даних, що реалізує інтерфейс ассоціативного масиву. Вона дозволяє зберігати пари (ключ-значення) і здійснювати операцію пошуку за ключем.

*К* — кількість ключів.

*h(k)* — хеш-функція, яка має не більше, ніж {M} різних значень, тобто

1. **Опис алгоритму**

*Ідеальна хеш-функція (англ. Perfect hash function)* — хеш-функція, яка без колізій відображає різні елементи з множини об'єктів на множину ключів за O (1) часу в гіршому випадку.

Для реалізації ідеального хешування побудовано дворівневу хеш-таблицю, в якій перший рівень звичайний, але в якості даних буде виступати ще одна хеш-таблиця в якій і будуть зберігатись ключі. Оскільки параметри таблиці другого рівня залежать від кількості в ній ключів, то можлива реалізація лише статичної хеш-таблиці (додавання і вилучення ключів неможливі).

*Перший рівень:*

Використовується функція . Тобто n ключів кодуються в m рядків. При цьому p має бути простим і перевищувати m та будь-який з ключів.

*Другий рівень:*

На даному рівні замість створення списку ключів будемо використовувати вторинну хеш-таблицю , що зберігає всі ключі.

Використовується така ж функція , але зі своїми коефіцієнтами. Функція вибирається з множини універсальних хеш-функцій .

Для забезпечення відсутності колізій та оптимізації значення встановлюється як квадрат кількості ключів в даній хеш-таблиці. При появі колізії функція перезапускається.

*(приклад хеш-таблиці)*

1. **Аналіз алгоритму**

О(1) у найгіршому випадку.

Очікувана загальна пам’ять під даний алгоритм

1. **Реалізація алгоритму**

Реалізовано на мові С# в середовищі розробки Microsoft [Visual Studio 2017](https://www.visualstudio.com/ru/).

Вибір мови зроблено з урахуванням підтримки Windows Forms.

1. **Розрахунки**

, n – кількість ключів.

1. **Висновки**

*Найпоширеніше використання ідеального хешування – перевірка наявності елемента в базі даних. Часто використовується в словниках. Через, порівняно, величезне використання пам’яті, статичність, використання ідеального хешування в інших сферах не рентабельне. Однак, він має надзвичайно швидкі обчислення та в ньому відсутні колізії, а отже, алгоритм може розвинутись в майбутньому.*

1. **Особливості реалізації даного варіанту**

Структурних відмінностей між даними варіантами не виявлено.

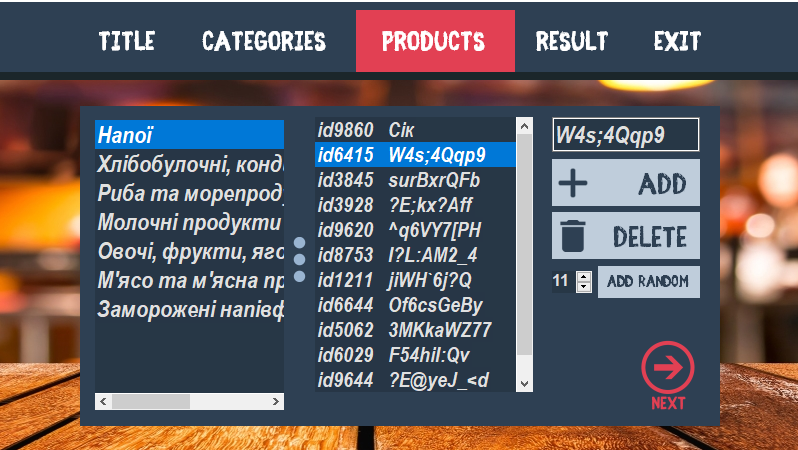
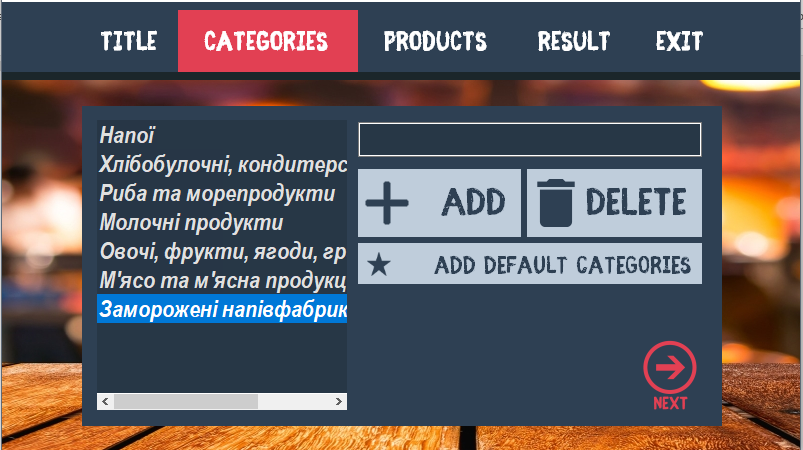
1. **Основні модулі програми**

Основна функція “static void Main()” в файлі Program.cs. Функціонал програми прив’язаний до подій з візуальними компонентами. Забезпечення всіх даних знаходиться в файлі main.cs. Реалізація алгоритму в файлі Algorithm.cs.

1. **Використані структури даних**

Для безпеки пам’яті в с# не існує вказівників \*. Однак можна і потрібно ініціалізувати структури, що і було зроблено. Всі дані були збережені в класах та їх масивах.

1. **Інтерфейс програми**



1. **Тестовий приклад**
2. Запустити програму. Можна відкрити Visual Studio або запустити файл Lab1.exe з папки /bin/Debug/..
3. Натиснути Next. Відкриється вкладка Categories.
4. Натиснути Add default categories. Натиснути Next. Відкриється вкладка Product.
5. Вибрати пункт Напої. Натиснути на Add random декілька разів. Натиснути Next. Відкриється вкладка Result.
6. Вибрати пункт Напої. Налаштування справа
7. Натиснути Exit або Next. Натиснути Exit.
8. **Використані джерела**
9. <https://ru.wikipedia.org/wiki/Хеширование>
10. <https://neerc.ifmo.ru/wiki/index.php?title=Идеальное_хеширование>
11. <https://ru.wikipedia.org/wiki/Хеш-таблица>
12. Т. Кормен. Алгоритми: побудова й аналіз. Третє видання (Розділ 11)
13. **GitHub**