УДК 004.422.63

Петров Д. Ю., здобувач вищої освіти,
Поремський Ю. В., канд. техн. наук,
старший викладач кафедри інформаційних технологій

ХЕШ-ТАБЛИЦІ: КОНЦЕПЦІЇ, РЕАЛІЗАЦІЯ ТА ПРИКЛАДИ ВИКОРИСТАННЯ У ПРОГРАМУВАННІ

Донецький національний університет імені Василя Стуса, м. Вінниця

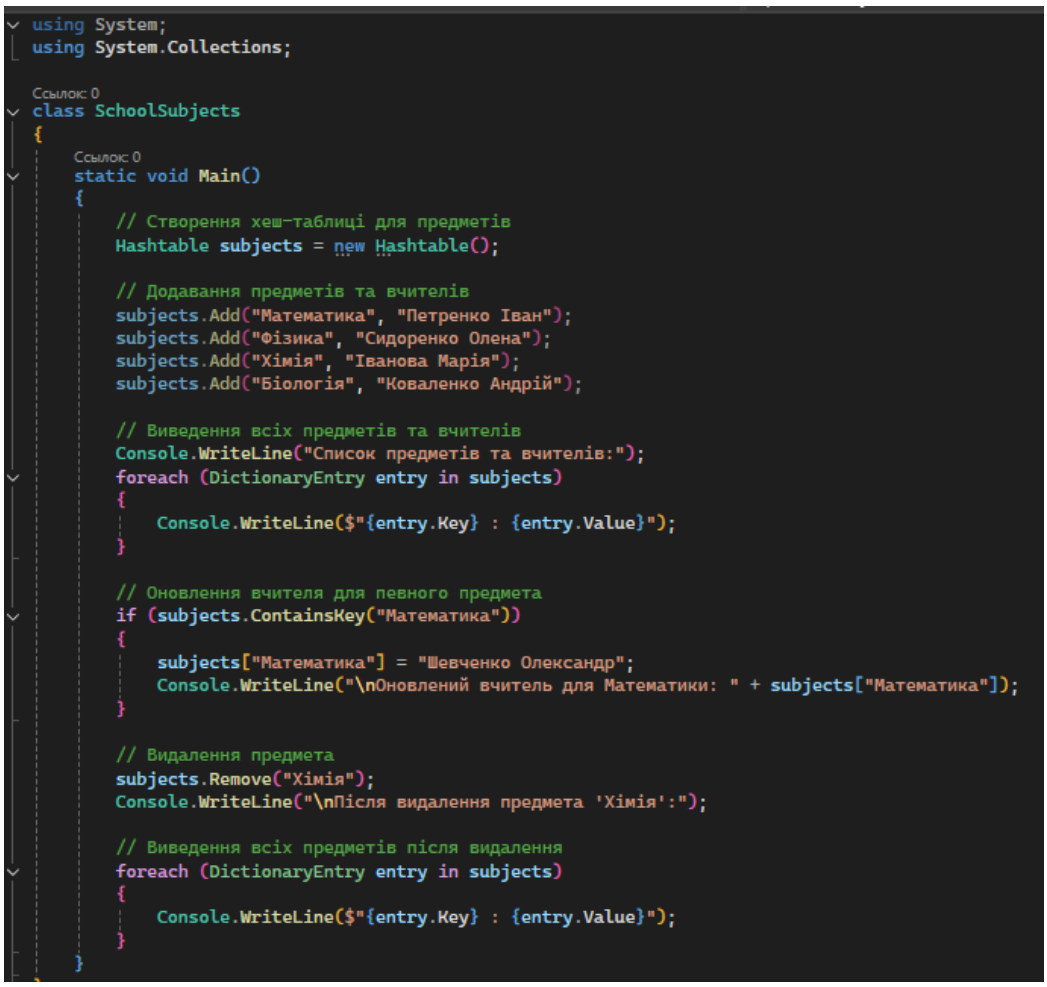
Хеш-таблиця – це структура даних, яка дає змогу швидко знаходити, вставляти і видаляти елементи з використанням хеш-функції для обчислення індексів у масиві або списку. Хеш-таблиці зазвичай використовуються для реалізації асоціативних масивів, також відомих як словники. Хеш-таблиці використовуються в багатьох сферах, включно з базами даних, кешуванням, пошуком дублікатів та обробкою великих обсягів інформації [1].

Основою роботи хеш-таблиць є хеш-функція – спеціальний алгоритм, який перетворює ключ у числовий індекс. Цей індекс використовується для швидкого доступу до елементу, що зберігається у відповідній позиції в масиві. Хеш-функція повинна бути ефективною, рівномірно розподіляти значення ключів по масиву та мінімізувати кількість конфліктів, що виникають у процесі обчислення. Завдяки використанню хеш-функції у багатьох випадках доступ до даних у хеш-таблиці здійснюється за постійний час, незалежно від кількості елементів [2].

Завдяки своїй універсальності хеш-таблиці широко застосовуються у різних областях інформатики та програмування. Вони є основою для реалізації словників, де кожен ключ відповідає певному значенню. До того ж хеш-таблиці використовуються для кешування даних, коли потрібно швидко зберігати і отримувати часто використовувану інформацію, як, наприклад, у вебкешах. Їх також активно застосовують у задачах аналізу текстів, зокрема для підрахунку частоти слів у тексті або для визначення унікальних слів у документі [3].

Аналогом хеш-таблиць у реальному житті можна вважати телефонні книги, де імена людей виступають у ролі ключів, а номери телефонів – значень. Є багато практичних прикладів хеш-таблиць, які використовуються в повсякденному житті. Популярним прикладом є бази даних імен користувачів і паролів. Щоразу, коли хтось реєструється на вебсайті за допомогою імені користувача та пароля, цю інформацію потрібно десь зберігати для подальшого пошуку. Хеш-функції широко використовуються в інформатиці та криптографії для різних цілей, зокрема і перевірки цілісності даних, зберігання паролів, цифрових підписів та індексації даних [4].

Приклад програмної реалізації простої хеш-таблиці для реалізації довідника предметів, які викладаються в школі, і відповідних вчителів наведено на рис. 1. Операції з додавання, оновлення, видалення предметів роблять код більш функціональним та реалістичним у контексті школи.



```
using System;
using System.Collections;

Ссылок: 0
class SchoolSubjects
{
    Ссылок: 0
    static void Main()
    {
        // Створення хеш-таблиці для предметів
        Hashtable subjects = new Hashtable();

        // Додавання предметів та вчителів
        subjects.Add("Математика", "Петренко Іван");
        subjects.Add("Фізика", "Сидоренко Олена");
        subjects.Add("Хімія", "Іванова Марія");
        subjects.Add("Біологія", "Коваленко Андрій");

        // Виведення всіх предметів та вчителів
        Console.WriteLine("Список предметів та вчителів:");
        foreach (DictionaryEntry entry in subjects)
        {
            Console.WriteLine($"{entry.Key} : {entry.Value}");
        }

        // Оновлення вчителя для певного предмета
        if (subjects.ContainsKey("Математика"))
        {
            subjects["Математика"] = "Шевченко Олександр";
            Console.WriteLine("\nОновлений вчитель для Математики: " + subjects["Математика"]);
        }

        // Видалення предмета
        subjects.Remove("Хімія");
        Console.WriteLine("\nПісля видалення предмета 'Хімія':");

        // Виведення всіх предметів після видалення
        foreach (DictionaryEntry entry in subjects)
        {
            Console.WriteLine($"{entry.Key} : {entry.Value}");
        }
    }
```

Рисунок 1 – Програмна реалізація хеш-таблиці

Розуміння принципів роботи хеш-таблиць є важливим складником знань програміста, оскільки ці структури даних дають змогу створювати ефективні алгоритми для роботи з великими наборами даних. Завдяки їм можна оптимізувати продуктивність систем, зменшити час виконання задач та забезпечити швидкий доступ до необхідної інформації [5].

Список використаних джерел

1. Думка Експерта. 2024. URL: https://luka.almedia.com.ua/ukraincyam/de-vikoristovuietsya-kheshuvannya-v-realnomu-zhitti.html (дата звернення 30.11.2024).

2. Чен С. Guru99. 2024. URL: https://www.guru99.com/uk/hash-table-data-structure.html (дата звернення 30.11.2024).

УДК 519.17:004.021

Радченко Р. П., здобувач вищої освіти,
Сеник І. О., асистент кафедри
інформаційних технологій

ГРАФИ ТА АЛГОРИТМИ ПОШУКУ: ТЕОРІЯ, ЗАСТОСУВАННЯ ТА ОПТИМІЗАЦІЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Графи є універсальною структурою даних, яка дає змогу моделювати широкий спектр реальних об’єктів та процесів, як-от соціальні мережі, транспортні системи, ієрархії. Завдяки своїй гнучкості графи використовуються для розв’язання задач, що виникають у різних галузях комп’ютерних наук.

Основу роботи з графами становлять алгоритми пошуку. Серед них найпоширенішими є пошук у ширину (BFS) та пошук у глибину (DFS). Ці алгоритми допомагають обійти всі вершини графу, визначити його зв’язність, знайти шляхи між заданими вершинами та вирішити інші базові задачі.

Для знаходження найкоротших шляхів у графах використовуються більш складні алгоритми, як-от алгоритм Дейкстри та алгоритм Беллмана–Форда. Алгоритм Дейкстри забезпечує ефективне знаходження найкоротших шляхів у графах з невід’ємними вагами, тоді як алгоритм Беллмана–Форда здатен працювати з графами, що містять від’ємні ваги.

Алгоритми на графах знаходять своє застосування в реальних системах. Наприклад, вони широко використовуються для оптимізації маршрутів у транспортних мережах, де важливо мінімізувати час або вартість переміщення. У соціальних мережах графові алгоритми дають змогу аналізувати зв’язки між користувачами, що допомагає у виявленні впливових осіб або побудові рекомендаційних систем. Важливою задачею є побудова мінімального кістякового дерева. Алгоритми Пріма та Крускала допомагають знайти дерево, яке з’єднує всі вершини графу з мінімальною загальною вагою. Це має практичне значення у проєктуванні мереж, наприклад, електричних чи комунікаційних.

Аналіз складності алгоритмів на графах є ключовим аспектом їх застосування. Залежно від розміру та властивостей графу обирають той чи інший алгоритм, який забезпечить оптимальний результат за прийнятний час.

Програма нижче демонструє базову реалізацію алгоритму BFS для графу, представленого у вигляді списку суміжності. Алгоритм проходить всі вершини графу і виводить порядок їх відвідування (рис. 1 та рис. 2).

Цей код демонструє просту реалізацію алгоритму BFS, який дає змогу проходити всі вершини графу від вказаної вершини. BFS широко використовується в задачах, що потребують знаходження найкоротшого шляху у незваженому графі або перевірки зв’язності графу [4].

Графи та алгоритми пошуку продовжують відігравати ключову роль у комп’ютерних науках і технологіях, забезпечуючи основи для ефективного вирішення складних задач у реальному світі.