**Державний вищий навчальний заклад  
Ужгородський національний університет  
Факультет інформаційних технологій**

ПРАКТИЧНА РОБОТА № 8  
**Тема:** Хеш-таблиці

Виконав студент  
І курсу спеціальності «Інженерія програмного забезпечення»

Романюк Артем

**Ужгород-2025**

**Мета:** познайомитися з хеш-функціями і хеш-таблицями та отримати навички програмування алгоритмів, що їх обробляють.

**Питання для самоконтролю**

1. Що таке асоціативний масив?
2. Яка структура даних називається хеш-таблицею? Які основні операції вона має підтримувати?
3. У чому полягає процес розв'язання колізій?
4. Які види хеш-таблиць бувають? Які відмінності між ними?
5. Яким чином реалізується процес видалення ключів у хеш-таблиці з відкритою адресацією?
6. Поясніть концепцію методу ланцюжків для розв’язання колізій при реалізації хеш-таблиць.

**Завдання до роботи**

1. Написати код програми, яка реалізує роботу з хеш-таблицею, відповідно до обраного варіанту. Програма повинна підтримувати мінімальні методи хеш-таблиць: додавання, видалення, пошук. Алгоритм хешування ключа студент обирає самостійно.
2. Оформити звіт та завантажити його в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
3. Підготувати відповіді на контрольні питання.

**Хід роботи:**

**Варіант 3**

| 3 | Реалізувати кеш на основі хеш-таблиці для збереження результатів запитів до зовнішнього API. Ключем буде URL запиту, а значенням - результат відповіді сервера. Перед виконанням кожного запиту перевірити, чи є результат в кеші. |
| --- | --- |

Реалізація в коді:

| class HashTable {  #size;  get size() { return this.#size };  #hashTable;  constructor(size) {  this.#size = size;  this.#hashTable = new Array(size).fill(null).map(() => []);  }  #hashFunction(key) {  const multiplier = 31;  let index = 0;  key = key.toString();  for (let i = 0; i < key.length; i++) {  index += key.charCodeAt(i) \* Math.pow(multiplier, i);  }  return index % this.#size;  }  add(key, value) {  let index = this.#hashFunction(key);  for (let pair of this.#hashTable[index]) {  if (pair[0] === key) {  pair[1] = value;  return;  }  }  this.#hashTable[index].push([key, value]);  }  remove(key) {  let index = this.#hashFunction(key);  for (let i = 0; i < this.#hashTable[index].length; i++) {  if (this.#hashTable[index][i][0] === key) this.#hashTable[index].splice(i, 1);  }  }  contains(key) {  let index = this.#hashFunction(key);  for (let pair of this.#hashTable[index]) {  if (pair[0] === key) return true  }  return false;  }  getValue(key) {  let index = this.#hashFunction(key);  for (let pair of this.#hashTable[index]) {  if (pair[0] === key) return pair[1];  }  }  clear() {  this.#hashTable = new Array(this.#size).fill(null).map(() => []);  }  print() {  console.log(this.#hashTable);  }  }  let cache = new HashTable(10);  async function getAlbum(albumId) {  const URL = `https://jsonplaceholder.typicode.com/albums?userId=${albumId}`;  if (cache.getValue(URL) !== undefined) {  console.log("Взято з кешу");  cache.getValue(URL).forEach(photo => {  console.log(`Номер: ${photo.id}, підпис: ${photo.title}`);  });  } else {  let album = await fetch(URL).then(album => album.json())  .catch(error => {  console.error(error);  return undefined;  });  if (album === undefined) return;  cache.add(URL, album);  album.forEach(photo => {  console.log(`Номер: ${photo.id}, підпис: ${photo.title}`);  });  }  }  (async () => {  await getAlbum(1);  console.log();  await getAlbum(1);  cache.print();  })(); |
| --- |

**Висновок:**

У ході виконання практичної роботи я ознайомився з принципами роботи хеш-таблиць, зокрема з реалізацією асоціативного масиву за допомогою хеш-функції та методу ланцюжків для розв’язання колізій. Було реалізовано кеш, який зберігає результати запитів до зовнішнього API, що дозволяє уникати повторних звернень до сервера при наявності збережених даних. Я навчився реалізовувати базові операції з хеш-таблицею — додавання, видалення, пошук та очищення. Отримані знання мають практичне застосування у розробці продуктивних програм, де важливе значення має швидкий доступ до даних.