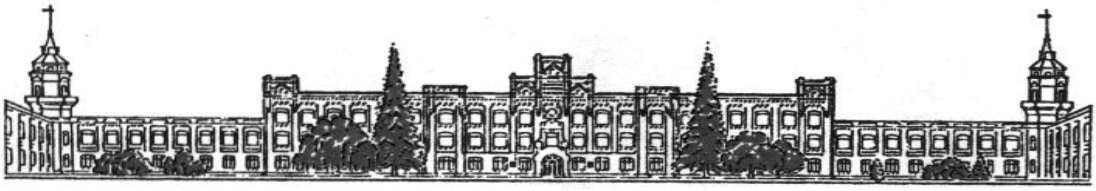

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра системного проектування

Алгоритми та структури даних

# Лабораторна робота №2

# “Дослідження структури даних хеш-таблиця”

Київ – 2021

## 1 Мета роботи

Ознайомитись і дослідити структуру даних хеш-таблиця. Набути навичок реалізації хеш-таблиці за методом ланцюжків мовою програмування C++, познайомитись з використанням STL контейнерів на прикладі unordered_map та порівняти власну реалізацію з готовим бібліотечним рішенням.

## 2 Короткі теоретичні відомості

- Генерація випадкових великих чисел:

Стандартна функція rand() з бібліотеки <cstdlib> генерує числа з максимальним значенням 32767, що для багатьох задач є недостатнім.

На базі цієї функції можна створити функцію для генерації необхідного за довжиною числа, випадково генеруючи кожну цифру цього числа з порозрядним заповненням. Для зберігання таких великих чисел замість звичайного типу даних ‘int’ потрібно використати тип ‘long long’. У даній лабораторній ці випадкові числа будуть використовуватись в якості ключів для хеш-таблиці, тому пропонуємо генерувати всі числа однакової довжини, наприклад 10-13. Враховувати, що в старшому розряді не може бути цифра 0.

- Хеш-таблиця:
  https://alextoolsblog.blogspot.com/2019/12/hash-map.html
  https://intellect.icu/khesh-tablitsa-kak-struktura-dannykh-4609
- Хеш-функції, загальна інформація:
  http://ruslan.rv.ua/python-essential/methodologies/hash/
  https://uk.wikipedia.org/wiki/Хеш-функція
- Яку хеш-функцію обрати?
  - Будь-яка хеш-функція з лекцій
  - Хеш-функція на основі ділення:
    hash(key) = key % m, де m - розмір масиву в хеш-таблиці.
  - Універсальна хеш-функція:
    hash(key) = ((a * key + b) % p) % m

де p - велике просте число, наприклад p = 9149658775000477, a, b - невеликі випадкові числа (a, b > 0), m - розмір масиву в хеш-таблиці.

**3 Завдання**

Написати програму для реалізації структури даних хеш-таблиця, яка дозволить проводити дуже швидкий пошук даних у наборі структур свого варіанту.

1. **Реалізувати функцію для генерації випадкового великого числа, що буде унікальним ключем-ідентифікатором даних у хеш-таблиці.**
2. **Реалізувати хеш-таблицю за методом ланцюжків:**

   2.1 Створити структуру Data для зберігання різнотипних даних відповідно до варіанту, вона буде використовуватись у якості значень для хеш-таблиці. Реалізувати генерацію випадкових даних для полів структури у пустому конструкторі Data().

   2.2 Створити структуру HashNode для зберігання ключа та значення

   2.3 Створити структуру LinkedList з підтримкою додавання елементів

   2.4 Створити структуру HashTable, яка буде містити масив зв'язних списків bucketsArray статичного розміру M = 10000 та реалізувати в ній основні функції:

   - hash(key) знаходження хеш-функції від ключа
   - insert(key, value) додавання значення під відповідним ключем (заміна у випадку існування ключа)
   - find(key) знаходження значення під відповідним ключем, функція повертає вказівник на знайдений елемент або NULL
   - erase(key) видалення значення під відповідним ключем
   - size() знаходження кількості елементів в хеш-таблиці

   Всі функції повинні працювати за O(1).

3. **Оптимізувати хеш-таблицю, додавши динамічну зміну розміру масиву bucketsArray в залежності від loadFactor (максимально можлива завантаженість таблиці):**

   3.1 Початковий розмір масиву bucketsArray встановити невеликим, наприклад m = 8, при перевищенні значення loadFactor динамічно збільшувати розмір (наприклад в 2 рази)

   3.2 Початкове значення loadFactor тимчасово обрати рівним 8.5

**4. Провести тестування, використавши вказану нижче функцію testHashTable(). Обрати оптимальне значення loadFactor. Перевірити правильність та швидкість роботи, порівнявши з готовим бібліотечним рішенням STL unordered_map.**

Необов'язкові завдання для отримання додаткових балів:

- Реалізувати ще одну хеш-таблицю, де в якості ключів використати тип string
- Реалізувати ще одну хеш-таблицю за методом відкритої адресації

Код для тестування хеш-таблиці:

```
#include <unordered_map>

using namespace std;

bool testHashTable()
{
    const int iters = 500000;
    const int keysAmount = iters * 1;

    // generate random keys:
    long long* keys = new long long[keysAmount];

    long long* keysToInsert = new long long[iters];
    long long* keysToErase = new long long[iters];
    long long* keysToFind = new long long[iters];

    for (int i = 0; i < keysAmount; i++)
    {
        keys[i] = generateRandLong();
    }
    for (int i = 0; i < iters; i++)
    {
        keysToInsert[i] = keys[generateRandLong() % keysAmount];
        keysToErase[i] = keys[generateRandLong() % keysAmount];
        keysToFind[i] = keys[generateRandLong() % keysAmount];
    }

    // test my HashTable:
    HashTable hashTable;

    clock_t myStart = clock();
    for (int i = 0; i < iters; i++)
    {
        hashTable.insert(keysToInsert[i], Data());
    }
    int myInsertSize = hashTable.size();
    for (int i = 0; i < iters; i++)
    {
        hashTable.erase(keysToErase[i]);
    }
    int myEraseSize = hashTable.size();
    int myFoundAmount = 0;
    for (int i = 0; i < iters; i++)
```

```
    {
        if (hashTable.find(keysToFind[i]) != NULL)
        {
            myFoundAmount++;
        }
    }
    clock_t myEnd = clock();
    float myTime = (float(myEnd - myStart)) / CLOCKS_PER_SEC;

    // test STL hash table:
    unordered_map<long long, Data> unorderedMap;

    clock_t stlStart = clock();
    for (int i = 0; i < iters; i++)
    {
        unorderedMap.insert({keysToInsert[i], Data()});
    }
    int stlInsertSize = unorderedMap.size();
    for (int i = 0; i < iters; i++)
    {
        unorderedMap.erase(keysToErase[i]);
    }
    int stlEraseSize = unorderedMap.size();
    int stlFoundAmount = 0;
    for (int i = 0; i < iters; i++)
    {
        if (unorderedMap.find(keysToFind[i]) != unorderedMap.end())
        {
            stlFoundAmount++;
        }
    }
    clock_t stlEnd = clock();
    float stlTime = (float(stlEnd - stlStart)) / CLOCKS_PER_SEC;

    cout << "My HashTable:" << endl;
    cout << "Time: " << myTime << ", size: " << myInsertSize << " - " << myEraseSize <<
", found amount: " << myFoundAmount << endl;
    cout << "STL unordered_map:" << endl;
    cout << "Time: " << stlTime << ", size: " << stlInsertSize << " - " << stlEraseSize
<< ", found amount: " << stlFoundAmount << endl << endl;

    delete keys;
    delete keysToInsert;
    delete keysToErase;
    delete keysToFind;

    if (myInsertSize == stlInsertSize && myEraseSize == stlEraseSize && myFoundAmount ==
stlFoundAmount)
    {
        cout << "The lab is completed" << endl;
        return true;
    }

    cerr << ":(" << endl;
    return false;
}
```

## 4 Зміст звіту

Звіт має містити:

1) Титульний аркуш
2) Мету роботи
3) Варіант завдання
4) Хід виконання роботи:
    a) Умова задачі
    b) Скріншоти результатів виконання
    c) Лістинг програми (код)
5) Висновки

## 5 Контрольні питання

1) Навіщо потрібна структура даних хеш-таблиця? Чим вона відрізняється від звичайного масива, списка? Коли її варто застосовувати?
2) Що таке хеш-функція та для чого вона потрібна? Що таке універсальне хешування?
3) Що таке колізії у хеш-таблицях? Які способи вирішення колізій існують, в чому переваги і недоліки кожного з них? В чому особливість реалізації методу відкритої адресації?
4) За що відповідає значення loadFactor та навіщо його використовувати?
5) Як залежить швидкість роботи хеш-таблиці від вибору хеш-функції та обраного значення loadFactor?

## 6 Варіанти завдань

### Варіант 1

Структура Гуртожиток має наступні поля: адреса, площа кімнати, наявність опалення. Створити студентський кампус в якому можна швидко знайти гуртожиток.

### Варіант 2

Структура Диплом має наступні поля: спеціальність, рік отримання, середній бал. Створити архів в якому можна швидко знайти диплом.

**Варіант 3**

Структура Викладач має наступні поля: ім’я, оцінка в кампусі, якість мікрофону. Створити кафедру на якій можна швидко знайти викладача.

**Варіант 4**

Структура Президент має наступні поля: ім’я, строк правління, площа палацу. Створити список інтерполу в якому можна швидко знайти президента.

**Варіант 5**

Структура Літак має наступні поля: авіакомпанія, вага, максимальна швидкість. Створити аеропорт в якому можна швидко знайти літак.

**Варіант 6**

Структура Корпус має наступні поля: адреса, номер, дотримання правил пожежної безпеки. Створити університет в якому можна швидко знайти корпус.

**Варіант 7**

Структура даних Програміст має наступні поля: посада, зарплата, кількість багів у коді. Створити галеру на якій можна швидко знайти програміста.

**Варіант 8**

Структура Пара з матаналізу має наступні поля: ім’я викладача, номер пари, кількість присутніх студентів. Створити розклад пар в якому можна швидко знайти пару з матаналізу.

**Варіант 9**

Структура Бомба має наступні поля: колір, вага, кількість тротилу в кг. Створити склад боєприпасів в якому можна швидко знайти бомбу.

**Варіант 10**

Структура Працівник деканату має наступні поля: посада, години роботи, продуктивність. Створити деканат в якому можна швидко знайти працівника.

**Варіант 11**

Структура Робот має наступні поля: ім’я, кількість рук, стать. Створити місто майбутнього в якому можна швидко знайти робота.

**Варіант 12**

Структура Кіт має наступні поля: кличка, кількість корму в день, пухнастість. Створити квартиру в якій можна швидко знайти кота.

**Варіант 13**

Структура Університет має наступні поля: назва, кількість студентів, наявність турнікетів. Створити місто в якому можна швидко знайти університет.

**Варіант 14**

Структура Додаткова сесія має наступні поля: причина, кількість незакритих предметів, шанс відрахування. Створити список неприємних речей в якому можна швидко знайти додаткову сесію.

**Варіант 15**

Структура Учень Хогвартсу має наступні поля: назва факультету, тип палички, вірогідність працевлаштування після навчання. Створити в якій можна швидко знайти учня Хогвартсу.

**Варіант 16**

Структура Контрольна робота має наступні поля: назва предмету, кількість завдань, ймовірність списування. Створити архів робіт в якому можна швидко знайти контрольну роботу.

**Варіант 17**

Структура Мем має наступні поля: назва шаблону, кумедність, кількість вподобайок. Створити паблік в якому можна швидко знайти мем.

**Варіант 18**

Структура iPhone має наступні поля: колір, номер моделі, наявність зарядного пристрою в комплекті. Створити магазин в якому можна швидко знайти iPhone.

**Варіант 19**

Структура Джедай має наступні поля: ім'я, кількість падаванів, колір світлового меча. Створити космічний корабель в якому можна швидко знайти джедая.

**Варіант 20**

Структура Лабораторна робота має наступні поля: назва предмету, номер, бажання виконувати. Створити пошту в якій можна швидко знайти лабораторну роботу.

**Варіант 21**

Структура Банківська картка має наступні поля: назва банку, номер, cvv2. Створити гаманець в якому можна швидко знайти банківську картку.

**Варіант 22**

Структура Алгоритм має наступні поля: назва, асимптотична складність, корисність. Створити лекцію в якій можна швидко знайти алгоритм.

**Варіант 23**

Структура Зомбі має наступні поля: ім'я, колір шкіри, кількість покусаних. Створити кладовище на якому можна швидко знайти зомбі.

**Варіант 24**

Структура даних Стікер має наступні поля: назва, популярність, кумедність. Створити пак в якому можна швидко знайти стікер.

**Варіант 25**

Структура Чашка має наступні поля: колір, діаметр, заповненість у см. Створити кухню на якій можна швидко знайти чашку.

**Варіант 26**

Структура Планета має наступні поля: назва, кількість супутників, наявність життя. Створити галактику в якій можна швидко знайти планету.

**Варіант 27**

Структура даних Чат має наступні поля: назва, кількість людей, частота лайливих слів. Створити соціальну мережу в якій можна швидко знайти чат.

**Варіант 28**

Структура Місто має наступні поля: назва, населення, наявність метро. Створити країну в якій можна швидко знайти місто.

**Варіант 29**

Структура TikTok відео має наступні поля: назва пісні, кількість вподобайок, тривалість ролику. Створити папку в якій можна швидко знайти TikTok відео.

**Варіант 30**

Структура Динозавр має наступні поля: назва виду, вміння літати, вага. Створити парк в якому можна швидко знайти динозавра.

**Варіант 31**

Структура Цукерка має наступні поля: виробник, вага, смак. Створити солодкий подарунок в якому можна швидко знайти цукерку.

**Варіант 32**

Структура Книга має наступні поля: назва, ціна, кількість сторінок. Створити бібліотеку в якій можна швидко знайти книгу.

**Варіант 33**

Структура даних Фільм має наступні поля: жанр, тривалість, оцінка. Створити сайт на якому можна швидко знайти фільм.

**Варіант 34**

Структура Гуртожиток має наступні поля: адреса, площа кімнати, наявність опалення. Створити студентський кампус в якому можна швидко знайти гуртожиток.

**Варіант 35**

Структура Диплом має наступні поля: спеціальність, рік отримання, середній бал. Створити архів в якому можна швидко знайти диплом.

**Варіант 36**

Структура Викладач має наступні поля: ім’я, оцінка в кампусі, якість мікрофону. Створити кафедру на якій можна швидко знайти викладача.

**Варіант 37**

Структура Президент має наступні поля: ім’я, строк правління, площа палацу. Створити список інтерполу в якому можна швидко знайти президента.

**Варіант 38**

Структура Літак має наступні поля: авіакомпанія, вага, максимальна швидкість. Створити аеропорт в якому можна швидко знайти літак.