

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Індивідуальне завдання № 3

з курсу «Алгоритми і структури даних»

(назва дисципліни)

на тему: «АТД»

Виконав: студент 2 курсу групи мф-21

напряму підготовки (спеціальності)

Комп'ютерні науки

122 Комп'ютерні науки

Єлагін І.А.

Прийняв: _____

Національна шкала: _____

Кількість балів: _____

Оцінка: ECTS _____

Завдання:

Є список запрошених людей на конференцію і є список тих, хто прибув. Перевірте, хто з запрошених дійсно приймає участь у конференції. Реалізуйте АДД множина (Set) за допомогою структури даних прямого доступу.

Код:

```
#include <iostream>

#include <vector>

using namespace std;

// Реалізація множини (Set) на основі хеш-таблиці
class HashSet {
private:
    static const int TABLE_SIZE = 100; // Розмір хеш-таблиці
    vector<vector<string>> table; // Хеш-таблиця: масив списків

    // Хеш-функція
    int hash(const string& key) const {
        int h = 0;
        for (char ch : key) {
            h = (h * 31 + ch) % TABLE_SIZE;
        }
        return h;
    }

public:
    HashSet() : table(TABLE_SIZE) {}

    // Додати елемент у множину
    void insert(const string& key) {
        int index = hash(key);
        for (const auto& element : table[index]) {
```

```

        if (element == key) {
            return; // Елемент вже існує
        }
    }
    table[index].push_back(key);
}

// Перевірити, чи існує елемент у множині
bool contains(const string& key) const {
    int index = hash(key);
    for (const auto& element : table[index]) {
        if (element == key) {
            return true;
        }
    }
    return false;
}

// Повернути всі елементи з множини
vector<string> getElements() const {
    vector<string> elements;
    for (const auto& bucket : table) {
        for (const auto& element : bucket) {
            elements.push_back(element);
        }
    }
    return elements;
}
};

int main() {
    // Список запрошених людей
    HashSet invited;
    invited.insert("Alice");

```

```

invited.insert("Bob");
invited.insert("Alastor");
invited.insert("Charlie");
invited.insert("Diana");

// Список тих, хто прибув
HashSet arrived;
arrived.insert("Bob");
arrived.insert("Charlie");
arrived.insert("Alastor");
arrived.insert("Eve");
arrived.insert("Frank");

// Визначаємо тих, хто дійсно бере участь у конференції
vector<string> attending;

for (const string& person : invited.getElements()) {
    if (arrived.contains(person)) {
        attending.push_back(person);
    }
}

// Виводимо список учасників
cout << "Conference participants:\n";
for (const auto& person : attending) {
    cout << person << '\n';
}

return 0;
}

```

Вхідні данні:

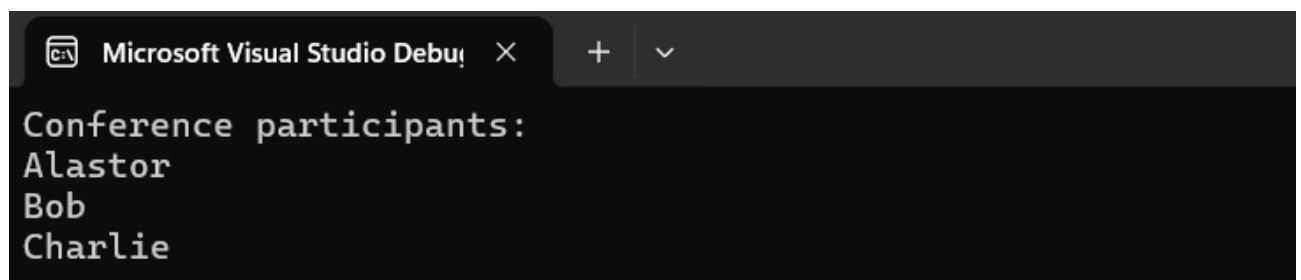
Створюємо дві множини і заповнюємо першу запрошеними людьми а другу тими хто прийшов.

```
// Список запрошених людей
HashSet invited;
invited.insert("Alice");
invited.insert("Bob");
invited.insert("Alastor");
invited.insert("Charlie");
invited.insert("Diana");

// Список тих, хто прибув
HashSet arrived;
arrived.insert("Bob");
arrived.insert("Charlie");
arrived.insert("Alastor");
arrived.insert("Eve");
arrived.insert("Frank");
```

Результат роботи коду:

У консолі ми побачимо список людей які були запрошені та прийшли на конференцію:



```
Conference participants:
Alastor
Bob
Charlie
```

Пояснення коду та операцій:

Клас HashSet:

Клас HashSet реалізує множину (set) на основі хеш-таблиці. Він дозволяє:

1. Зберігати унікальні елементи (рядки).
2. Виконувати операції додавання (insert), перевірки наявності (contains) та отримання всіх елементів (getElements).
3. Використовувати хешування для ефективної організації даних.

Поля класу:

1. TABLE_SIZE:
 - 1.1. Константа, що визначає розмір хеш-таблиці (100 комірок).
 - 1.2. Таблиця зберігає елементи у вигляді вектора списків, щоб обробляти колізії.
2. table:
 - 2.1. Вектор векторів vector<vector<string>>. Це хеш-таблиця, де кожна комірка (bucket) містить список (вектор) елементів з однаковим хешем. Колізії (коли два рядки мають однаковий хеш) вирішуються шляхом додавання елементів у список (ланцюжковий метод).

Конструктор HashSet()

1. Ініціалізує хеш-таблицю розміром TABLE_SIZE (100).
2. Кожна комірка таблиці спочатку є порожнім вектором.

Хеш-функція int hash(const string& key) const

Функція обчислює індекс для рядка key, щоб визначити, до якого "відра" (комірки) його зберігати.

Алгоритм:

1. Ініціалізує змінну h як 0.
2. Проходить по кожному символу рядка key:
 - 2.1. Оновлює h за формулою:
 - 2.1.1. 31 — просте число, часто використовується у хешуванні для мінімізації колізій.
 - 2.1.2. % TABLE_SIZE гарантує, що індекс буде в межах [0, 99].
3. Повертає отриманий індекс.

Метод void insert(const string& key)

Додає рядок key до множини, якщо його ще немає.

Алгоритм:

1. Обчислює індекс за допомогою hash(key).
2. Перевіряє, чи вже є рядок у відповідному "відрі" (векторі table[index]):
 - 2.1. Якщо рядок є, виходить із функції (return).
3. Якщо рядок не знайдено, додає його до вектора table[index].

Метод bool contains(const string& key) const

Перевіряє, чи є рядок key у множині.

Алгоритм:

1. Обчислює індекс за допомогою hash(key).
2. Перебирає всі елементи у векторі table[index]:
 - 2.1. Якщо знаходить збіг (element == key), повертає true.
3. Якщо переглянуто всі елементи, повертає false.

Метод vector<string> getElements() const

Повертає всі елементи множини у вигляді одного вектора.

Алгоритм:

1. Створює порожній вектор elements.

2. Перебирає всі комірки таблиці (table).
3. Для кожної комірки:
 - 3.1. Перебирає всі елементи у векторі цієї комірки.
 - 3.2. Додає їх до elements

Головна функція: int main()

Етапи роботи:

1. Створення множин invited і arrived:
 - 1.1. Заповнюються списки запрошених та прибулих за допомогою методу insert.
2. Отримання списку учасників конференції:
 - 2.1. Використовуємо invited.getElements(), щоб отримати всі елементи множини invited.
 - 2.2. Для кожного елемента перевіряємо, чи є він у множині arrived за допомогою arrived.contains(person).
 - 2.3. Якщо елемент знайдено, додаємо його до вектора attending.
3. Виведення результатів:
 - 3.1. Виводимо список учасників конференції, перебираючи вектор attending.