TD3: STD-LIB

C++ - ENSISA 2A

Ali El Hadi ISMAIL FAWAZ

November 14, 2024

I- Vector

Un 'vector' en C++ est simplement un dynamic array, on peut ajouter/enlever des éléments durant l'exécution.

```cpp
#include <vector> // librarie de vector
#include <algorithm> // librairie des fonctions sur le vector
#include <iostream>

using namespace std;

int main(){

    // Declaration d'un vector de type int
    vector<int> v1;

    // Declatation et initialization d'un vector de type double
    vector<double> v2(5, 10.5); // 5 elements de valeur 10.5
    vector<double> v3 = {1.2,2.1,3.2,4.3};

    // Vector 2D de 4 lignes et 2 colonnes, elements = 0
    vector<vector<int>> v2D(4, vector<int> (2,0));

    // Ajouter un element a la fin
    v1.push_back(10);
    v1.push_back(1);
    v1.push_back(2);
    v1.push_back(4);
    cout << v1[0] << endl;

    // Taille de vector
    int v1_size = v1.size();
    cout << v1_size << endl;

    // Iteration sur vector
    // Avec des indices
    for(int i = 0; i < v1_size; i++){
    cout << v1[i] << endl;
    }

    // Avec des pointeurs
    for(auto it = v1.begin(); it != v1.end(); it++){
    cout << *it << endl;
    }

    // Avec de Range-Based boucle for
    for(int value : v1){
    cout << value << endl;
    }

    // Enlever le derner element
    v1.pop_back();

    // Ajouter un element sur un indice specifique
    v1.insert(v1.begin()+1, 7);

    // Enlever un element sur un indice
    v1.erase(v1.begin()+1);

    // Ordonner les elements
    sort(v1.begin(), v1.end());
}
```

Plus d'info sur les 'vector', visiter geeksforgeeks

Autre outils de librarie standard

- 'set': c'est un 'container' qui contient des éléments unique et ordonnés
- 'unordered_set': comme 'set' mais non ordonné
- 'map': c'est une carte qui contient des clés et une valeur pour chaque clé. Les clés sont ordonnées
- 'unordered_map': comme 'map' mais les clés sont pas ordonnées.
- 'queue': First In First Out (FIFO)
- 'stack': Last In First Out (LIFO)

Exercice I: MAP

Vous disposez d'un vecteur de valeurs entières.

- Implémentez les deux fonctions countFrequencyBruteForce et countFrequencyOptimal pour obtenir la fréquence de chaque élément.

countFrequencyBruteForce est une fonction de type void, elle affichera les fréquences à l'intérieur.
countFrequencyOptimal est une fonction de type map¡int, int¿, elle renverra la map des fréquences.
Comparez les complexités.

```
#include <iostream>
#include <vector>
#include <map>

using namespace std;

void countFrequencyBruteForce(const vector<int>& numbers) {
    // Votre code
}

map<int, int> countFrequencyOptimal(const vector<int>& numbers) {
    // Votre code
}

int main() {
    vector<int> numbers = {1, 2, 3, 2, 4, 1, 5, 5, 6};

    // Test countFrequencyBruteForce
    cout << "Frequency (Brute Force):" << endl;
    countFrequencyBruteForce(numbers);

    // Test countFrequencyOptimal
    cout << "\nFrequency (Optimal):" << endl;
    map<int, int> frequencyMapOptimal = countFrequencyOptimal(numbers);
    for (const auto& entry : frequencyMapOptimal) {
        cout << entry.first << ": " << entry.second << " times" << endl;
    }

    return 0;
}
```

Exercice II: UNORDERED MAP

Vous disposez d'un vecteur nums de valeurs entières et d'une variable entière target. Implémentez les deux fonctions twoSumBruteForce et twoSumOptimal pour retourner un vecteur d'entiers de 2 éléments, contenant les indices de nums où la somme des deux éléments indexés donne la valeur de la variable target.

```
#include <iostream>
#include <vector>
#include <unordered_map>

using namespace std;

vector<int> twoSumBruteForce(const vector<int>& nums, int target) {
    // Votre code
}

vector<int> twoSumOptimal(const vector<int>& nums, int target) {
    // Votre code
}

int main() {
    vector<int> nums = {2, 7, 11, 15};
    int target = 9;

    vector<int> indicesBruteForce = twoSumBruteForce(nums, target);
    cout << "Brute Force Solution: ["
         << indicesBruteForce[0]
         << ", "
         << indicesBruteForce[1]
         << "]"
         << endl;

    vector<int> indicesOptimal = twoSumOptimal(nums, target);
    cout << "Optimal Solution: ["
         << indicesOptimal[0]
         << ", "
         << indicesOptimal[1]
         << "]"
         << endl;

    return 0;
}
```

Exercise III: QUEUE & STACK

Vous disposez d'une variable de type 'string'. Implémentez la fonction isPalindrome qui retourne true si la chaîne est un palindrome et false sinon.

Un palindrome signifie qu'il se lit de la même manière de gauche à droite et de droite à gauche, formant ainsi un mot ou une phrase symétrique. Par exemple, les mots "radar" et "été" sont des palindromes.

```
#include <iostream>
#include <queue>
#include <stack>
#include <cctype>

using namespace std;

bool isPalindrome(const string& input) {
    // Votre code
}
```

```
int main() {
    cout << boolalpha;

    cout << "Is 'racecar' a palindrome? "
         << isPalindrome("racecar") << endl;

    cout << "Is 'hello' a palindrome? "
         << isPalindrome("hello") << endl;

    return 0;
}
```