Data Structures & Algorithms – Cấu Trúc Dữ Liệu & Giải Thuật

Nguyễn Quản Bá Hồng*

Ngày 1 tháng 6 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Data Structures & Algorithms – Cấu Trúc Dữ Liệu & Giải Thuật.*
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Vector in C++ STL

Resources – Tài nguyên.

1. Geeks4Geeks/vector in C++ STL

Definition 1. C++ vector *is a dynamic array that stores collection of elements of same type in contiguous memory. It has the ability to resize itself automatically when an element is inserted or deleted.*

– C++ vector *là một mảng động lưu trữ tập hợp các phần tử cùng loại trong bộ nhớ liền kề. Nó có khả năng tự động thay đổi kích thước khi một phần tử được chèn vào hoặc xóa.*

1.1 Create vectors – Tạo vectors

Before creating a vector, we must know that a vector is defined as the `std::vector` class template in the `<vector>` header file.

```
vector<T> v;
```

where `T` is the type of elements & `v` is the name assigned to the vector.

Now we are creating an instance of `std::vector` class. This requires us to provide the type of elements as template parameter.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    // Creating an empty vector
    vector<int> v1;
    return 0;
}
```

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

We can also provide the values to be stored in the vector inside `{}` curly braces. This process is called *initialization*.

```
#include <bits/stdc++.h>
using namespace std;
void printVector(vector<int>& v) {
    for (auto x: v) {
        cout << x << " ";
    }
    cout << endl;
}

int main() {
    // Creating a vector of 5 elements from initializer list
    vector<int> v1 = {1, 4, 2, 3, 5};

    // Creating a vector of 5 elements with default value
        vector<int> v2(5, 9);

    printVector(v1);
    printVector(v2);
        return 0;
}
```

Output:

```
1 4 2 3 5
9 9 9 9 9
```

Statement `vector<int> v1 = {1, 4, 2, 3, 5}` initializes a vector with given values. Statement `vector<int> v2(5, 9)` creates a vector of size 5 where each element initialized to 9.

Remark 1. *Statement* `vector<int> v=`$\{v_1, v_2, \ldots, v_n\}$ *initializes a vector with given values. Statement* `vector<int> v(n, a)` *creates a vector of size* $n \in \mathbb{N}^\star$ *where each element initialized to* $a \in \mathbb{Z}$.

1.2 Insert elements – Chèn phần tử

An element can be inserted into a vector using vector insert() method which takes linear time. But for the insertion at the end, the vector `push_back()` method can be used, which is much faster, taking only constant time.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<char> v = {'a', 'f', 'd'};

    // Inserting 'z' at the back
    v.push_back('z');

    // Inserting 'c' at index 1
    v.insert(v.begin() + 1, 'c');

    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";
    return 0;
}
```

Output: `a c f d z`.

1.3 Access or update elements – Tiếp cận hoặc cập nhật các phần tử

Just like arrays, vector elements can be accessed using their index inside the [] subscript operator. While accessing elements, we can also update the value of that index using assignment operator =. The `[] subscript operator` doesn't check whether the given index exists in the vector or not. So, there is another member method vector al() for safely accessing or update elements.

```
#include <bits/stdc++.h>
using namespace std;

```

```
int main() {
    vector<char> v = {'a', 'c', 'f', 'd', 'z'};

        // accessing & printing values
    cout << v[3] << endl;
    cout << v.at(2) << endl;

    // updating values using indexes 3 & 2
    v[3] = 'D';
    v.at(2) = 'F';

    cout << v[3] << endl;
    cout << v.at(2);
    return 0;
}
```

1.4 Find vector size – Tìm cỡ/kích thước của vector

1 of the common problems with arrays was to keep a separate variable to store the size information. Vector provides the solution to this problem by providing size() method.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<char> v = {'a', 'c', 'f', 'd', 'z'};

    // finding size
    cout << v.size();
    return 0;
}
```

Output: 5.

1.5 Traverse vector – Duyệt vector

Vector in C++ can be traversed using indexes in a loop. The indexes start from 0 & go up to a vector size −1. To iterate through this range, we can use a loop & determine the size of the vector using the vector size() method.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<char> v = {'a', 'c', 'f', 'd', 'z'};

    // traversing vector using range based for loop
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";
    return 0;
}
```

Output: `a c f d z`.

We can also use a range-based loop for simple traversal.

1.6 Delete elements – Xóa phần tử

An element can be deleted from a vector using vector erase() but this method needs iterator to the element to be deleted. If only the value of the element is known, then `find()` function is used to find the position of this element.

For the deletion at the end, the vector `pop_back()` method can be used, & it is much faster, taking only constant time.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<char> v = {'a', 'c', 'f', 'd', 'z'};

    // deleting last element 'z'
    v.pop_back();

    // deleting element 'f'
    v.erase(find(v.begin(), v.end(), 'f'));

    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    return 0;
}
```

Output:

```
a c f d
a c d
```

1.7 Other operations – Các thao tác khác

Vector is 1 of the most frequently used containers in C++. It is used in many situations for different purposes. The following examples aim to help you master vector operations beyond the basics.

2 Miscellaneous