# Các container trong C++

# I. Vector trong C++

Vector trong C++ giống dynamic array (mảng động) nhưng có khả năng tự động thay đổi kích thước khi một phần tử được chèn hoặc xóa tùy thuộc vào nhu cầu của tác vụ được thực thi, với việc lưu trữ của chúng sẽ được vùng chứa tự động xử lý. Các phần tử vector được đặt trong contiguous storage (bộ nhớ liền kề) để chúng có thể được truy cập và duyệt qua bằng cách sử dụng iterator.

1.1 Ưu điểm của Vector

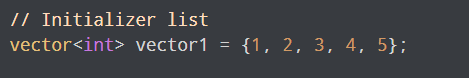
* Ta không cần phải khai báo kích thước của mảng ví dụ int A[100]…, vì vector có thể tự động nâng kích thước lên.
* Nếu ta thêm 1 phần tử vào vector đã đầy rồi, thì nó sẽ tự động tăng kích thước của nó lên để dành chỗ cho giá trị mới này.
* Dùng số phần tử âm vẫn được trong vector ví dụ A[-6], A[-9], rất tiện trong việc cài đặt các giải thuật.
* **Cơ chế ngăn chặn rò rỉ bộ nhớ của Vector:** Khi một biến **vector** rời khỏi phạm vi đoạn code mà chương trình đang chạy, nó sẽ tự động giải phóng những phần bộ nhớ mà nó kiểm soát (nếu cần). Điều này không chỉ tiện dụng (vì bạn không cần tự tay giải phóng bộ nhớ), mà nó còn giúp ngăn ngừa lỗi rò rỉ bộ nhớ (memory leaks).

**1.2 Sử dụng Vector**

Khai báo: Thư viện vector không cần cung cấp độ dài mảng tại thời điểm biên dịch.

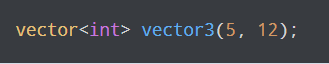
Có 2 cách khởi tạo 1 biến vector

* Cách 1:



Vector1 có giá trị khởi tạo là 1 2 3 4 5

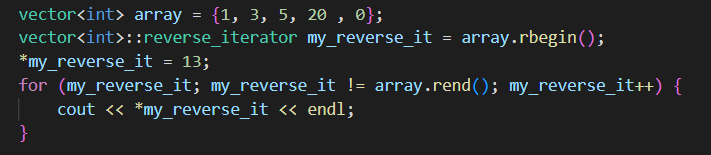
* Cách 2:



Vector3 có size là 5 và tất cả 5 phần tử có giá trị 12

Các hàm có sẵn với vector

* size(): lấy ra kích thước của mảng
* resize(): Thay đổi kích thước của mảng
* push\_back(): Thêm phần tử vào cuối vector
* pop\_back(): Xóa phần tử ở cuối vector
* at(int n): Truy cập vào phần tử thứ n của vector – Có thể cách khác là my\_vector[n]
* end(): Trả về vị trí đứng sau vị trí cuối cùng của vector
  + Ta hay sử dụng: vector<int>::iterator my\_it =  array.end() - 1; để truy cập vị trí cuối cùng của vector
* begin(): Trả về vị trí đầu tiên của vector
* rbegin(): Trả về vị trí cuối cùng của vector, được sử dụng với reverse\_iterator
* rend(): Trả về vị trí sau vị đầu tiên của vector, được sử dụng với reverse\_iterator



Kết quả:

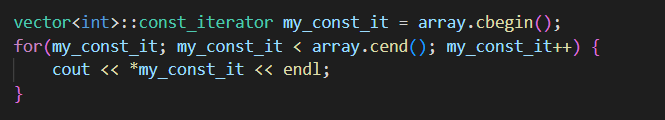


Chú ý: Tuy là đảo lộn giữa rbegin và rend, nhưng các thao tác của nó vẫn giống như bình thường

Ta thấy ở ví dụ trên ta sử dụng rbegin() và trỏ đến vị trí cuối nma vẫn sử dụng toán tử ++ như begin().

* cbegin(): Trả về 1 const\_iterator  trỏ vào vị trí đầu tiên của vector
* cend(): Trả về 1 const\_iterator  sau vị trí cuối cùng của vector

VD:



Kết quả:



* empty(): Trả về true nếu vector rỗng
* insert(iterator, index): Chèn một giá trị index vào vị trí trước vị trí iterator đang trỏ tới.
* earse(): Xóa 1 vị trí tại vị trí iterator đang chỉ tới
* front(): Trả về giá trị đầu tiên trong vector