**Vector và các khái niệm cơ bản**

1. **Khái niệm**

-Để giải quyết một bài toán quản lý danh sách khách hàng hay quản lý một thư viện, trường học, bạn có thể dùng nhiều cách như: sử dụng mảng, sử dụng danh sách liên kết, queue, stack, … Nếu bạn sử dụng các phương pháp thông thường trong 1 bài tập, chuyện này sẽ rất dễ. Tuy nhiên, đối với những bài toán có sự biến động nhiều hơn trong mảng, Vector C++ sẽ là phương pháp thích hợp nhất.

-Vector C++ là một mảng có khả năng “động” khá giống với dynamic array. Tuy nhiên, Vector C++ có khả năng tự động thay đổi kích thước nếu một phần tử bị xóa hay được chèn vào thêm. Đồng nghĩa với việc vùng chứa sẽ tự xử lý việc lưu trữ.

-Các phần tử Vector C++ sẽ được đặt trong một bộ nhớ liền kề hay contiguous storage và bạn có thể sử dụng iterator để truy cập và duyệt chúng.

1. **Cách khai báo**

-Khai báo thư viện **#include<Vector>**

-Chúng ta sẽ có 3 cách để khởi tạo chính bao gồm:

1. Dùng hàm push\_back()

Nếu bạn muốn các phần tử được lưu trữ trong Vector được đẩy lùi về sau từng phần tử một, bạn có thể sử dụng push\_back()

**Cấu trúc: Vector\_name.push\_back(element\_value);**

1. Sử dụng overload constructor của Vector Class

Cấu trúc: **Vector<object\_type>Vector\_name(number\_of\_repetition,element\_value);**

1. Sử dụng Array

Cấu trúc:

**Vector<object\_type> Vector\_name {val1,val2,val3,....,valn};**

1. **Các hàm trong vector**:
   1. Các hàm chức năng thay đổi 1 dối tượng vector (Modifiers)

* **push\_back():** như trong ví dụ ở trên, hàm này được sử dụng để đẩy một phần tử trong Vector về sau. Nếu kiểu đối tượng được truyền trong tham số không giống với kiểu của Vector thì chúng sẽ bị ném bỏ.
* **assign():** ghi đè các giá trị mới bằng cách thay thế giá trị cũ
* **pop\_ back():** sử dụng để làm giảm kích thước của Vector xuống 1 phần tử.
* **insert():** sử dụng để chèn phần tử mới trước vị trí được trỏ bởi vòng lặp
* **erase():** sử dụng để xóa các phần tử khỏi vòng sp.
* **swap():** dùng để hoán đổi giá trị/ nội dung của Vector có cùng kiểu, không nhất thiết cùng kích thước.
* **clear():** dùng để loại bỏ các phần tử của vùng chứa Vector

Cấu trúc: **VectorName[pos].FunctionName ;**

* 1. Iterator
* begin(): Được sử dụng để trả về một trình lặp trỏ đến phần tử đầu tiên trong Vector.
* end(): sử dụng để trả về trình lặp đến phần tử cuối cùng trong Vector.

Vd:

|  |
| --- |
|  |

-Kết quả đầu ra chúng ta có là:

|  |
| --- |
|  |

Trong ví dụ này, bạn có thể thấy rằng chúng tôi sử dụng hàm begin() và end(). Đầu tiên, chúng tôi tạo vec1, sau đó đẩy lùi giá trị các giá trị từ 1 – 10 bằng vòng lặp for. Sau đó, chúng tôi in các giá trị của Vector bằng cách sử dụng vòng lặp for và sử dụng hàm **begin()** và **end()** để chỉ định điểm đầu cuối của vòng lặp.

* 1. Các hàm Capacity:
* **size()**: hàm này trả về số lượng phần tử trong Vector.
* **max\_size():** hàm sử dụng để trả về số phần tử tối đa Vector có thể giữ được.
* **capacity():** hàm được dùng để trả về kích thước không gian lưu trữ của Vector được cấp bằng đồ thị số.
* **resize():** hàm được sử dụng để chứa các phần tử “n”. Nếu kích thước hiện tại của Vector lớn hơn n, các phần tử phía sau n sẽ bị xóa khỏi Vector và ID các phần tử bổ sung sẽ được chèn sau Vector
* **empty():** nếu giá trị trả về của hàm là true, Vector của bạn đang trống. Nếu giá trị trả về là false, Vector của bạn không trống.

Cấu trúc: **VectorName.FunctionName ;**