|  |  |
| --- | --- |
|  | ***ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH***  ***TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN*** |

***Cấu trúc dữ liệu và giải thuật***

***Đề tài***

***TÌM HIỂU VỀ THƯ VIỆN VECTOR-C++***

* *Giáo viên hướng dẫn: Phan Thị Phương Uyên*
* *Võ Thế Minh – 18120211*
* *18CTT2 - LớpA - CTDL&GT*
* *Học thực hành ca2 sáng T5*

**Phần 1: Khái niệm**

1. **Vector**

* Vector là các ô chứa dữ liệu liên tiếp đại diện cho các mảng có thể thay đổi kích thước của mình.
* Giống như mảng, vector sử dụng các ô nhớ liên tiếp để lưu các phần tử của nó, tức là các phần tử của nó có thể được truy xuất qua các offset trên các con trỏ trỏ tới các phần tử của chúng với độ hiệu quả như với mảng. Nhưng vector khác mảng ở chỗ là nó có kích thước động, với khả năng lưu trữ được xử lý tự động bởi các ô chứa.

1. **Tính hiệu quả của vector so với các cấu trúc khác:**

* So sánh với mảng thì vector sử dụng nhiều bộ nhớ hơn, bù lại cho khả năng quản lý vùng nhớ và tăng kích cỡ động một cách hiệu quả.
* So sánh với các kiểu dữ liệu động khác (như list...), vector rất hiệu quả trong việc truy xuất phần tử (như mảng) và tương đối hiệu quả với việc thêm hoặc xóa phần tử từ cuối. Với những hành động bao gồm việc chèn hoặc xóa các phần tử không phải ở cuối thì vector tệ hơn các kiểu dữ liệu động khác.

1. **Cách vector quản lí các phần tử:**

* Sâu bên trong thì vector dùng mảng được cấp phát động để lưu trữ các phần tử. Mảng này có thể cần phải tái cấp phát để có thể tăng kích thước khi có các phần tử mới được thêm vào. Tái cấp phát tức là xin cấp phát một mảng mới và chuyển toàn bộ phần tử tới mảng mới. Đây là một tác vụ cực kì tốn kém về mặt thời gian xử lý và vì thế vector không tái cấp phát mỗi lần có phần tử mới được thêm vào.
* Thay vào đó, các ô chứa của vector sẽ cấp phát dư ra một lượng bộ nhớ để đề phòng việc tăng lượng phần tử cần lưu trữ, vì vậy ô chứa của vector có lẽ sẽ có kích thước lớn hơn kích thước thực cần để lưu trữ các phần tử. Các thư viện có thể cài đặt nhiều cơ chế khác nhau cho việc tăng kích thước để cân bằng giữa bộ nhớ sử dụng và việc tái cấp phát.
* Nhưng trong bất kì trường hợp nào, chỉ nên tái cấp phát mỗi khi đạt một khoảng kích cỡ tăng theo logarit (logarit tự nhiên) để đảm bảo việc thêm các phần tử riêng lẻ vào cuối vector có độ phức tạp thời gian được khấu hao (giảm dần qua các lần).

**PHẦN 2: Các hàm trong vector**

|  |  |  |
| --- | --- | --- |
| TYPE | Tên hàm | Chức năng |
| Iterator | begin() | Trả về một iterator trỏ đến phần tử đầu tiên của vector |
| end() | Trả về một iterator trỏ đến phần tử cuối cùng của vector |
| rbegin() | Trả về một iterator trỏ đến phần tử đầu tiên của vector đã bị đảo ngược (ứng với phần tử cuối cùng của dãy ban đầu) |
| rend() | Trả về một iterator trỏ đến phần tử cuối cùng của vector đã bị đảo ngược (ứng với phần tử đầu tiên của dãy ban đầu) |
| cbegin() | Trả về một hằng iterator trỏ đến phần tử đầu tiên của vector |
| cend() | Trả về một hằng iterator trỏ đến phần tử cuối cùng của vector |
| crbegin() | Trả về một hằng iterator trỏ đến phần tử đầu tiên của vector đã bị đảo ngược (ứng với phần tử cuối cùng của dãy ban đầu) |
| crend() | Trả về một hằng iterator trỏ đến phần tử cuối cùng của vector đã bị đảo ngược (ứng với phần tử đầu tiên của dãy ban đầu) |
| Capacity | size() | Trả về số lượng phần tử có trong vector |
| max\_size() | Trả về số lượng phần tử tối đa mà vector có thể lưu trữ |
| capactity() | Trả về số lượng phần tử được cấp phát cho vector |
| resize(int n) | Đặt lại kích thước vector chứa n phần tử |
| empty() | Trả về false nếu vector trống, ngược lại, trả về true |
| shrink\_to\_fit() | Giải phóng các vùng nhớ dư thừa của vector |
| reserve(int n) | Yêu cầu cấp phát tổi thiểu để có thể chứa được n phần tử |
| Element access | Operator [int n] | Truy cập đến phần tử thứ n của vector và trả về tham chiếu đến phần tử đó |
| at(int n) |  |
| front() | Trả về tham chiếu tới phần tử đầu tiên của vector |
| last() | Trả về tham chiếu tới phần tử cuối cùng của vector |
| data() | Trả về con trỏ, trỏ đến vùng nhớ lưu trữ các phần tử của vector |
| Modifiers | assign() | Gán giá trị mới cho các phần tử trong vector bằng cách thay thế các phần tử cũ |
| push\_back() | Thêm phần tử vào vector |
| pop\_back() | Lấy ra phần tử cuối cùng của vector |
| insert() | Chèn phần tử mới vào trước phần tử được được chỉ định |
| clear() | Loại bỏ tất cả các phần tử ra khỏi vector |
| emplace() | Mở rộng bộ nhớ của vector bằng cách chèn thêm phần tử vào vị trí được truyền tới |
| emplace\_back() | Mở rộng bộ nhớ của vector bằng cách chèn thêm phần tử vào cuối vector |
| erase() | Loại bỏ phần tử hoặc nhóm phần tử được chỉ định ra khỏi vector |
| swap() | Hoán vị các phần tử của vector gọi hàm với vector truyền vào (có cùng kiểu dữ liệu). Số lượng phần tử của 2 vector có thể khác nhau |

**PHẦN 3: Các nguồn tham khảo**

<http://www.cplusplus.com/reference/vector/vector/>

* Kết thúc