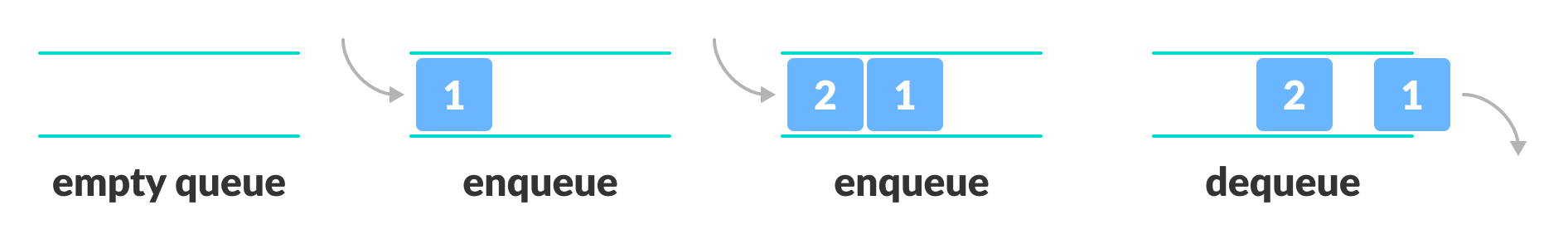
1. **QUEUE (Hàng đợi 2 đầu)**
2. **Queue là gì**

* Tương tự như Stack, Queue (hàng đợi) cũng là một cấu trúc dữ liệu. Queue tuân theo quy tắc First in first out(nhập trước xuất trước) - FIFO:
* Theo thuật ngữ lập trình, việc đưa các mục vào hàng đợi được gọi là **enqueue** và việc xóa các mục khỏi hàng đợi được gọi là **dequeue** .

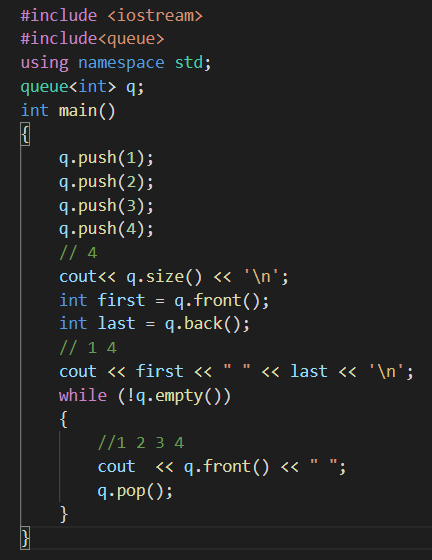


1. **Các thao tác trên queue**
   * **push()**: Thêm một phần tử vào cuối hàng đợi
   * **pop()**: Xóa 1 phần tử khỏi đầu hàng đợi
   * **front()**: Trả về phần tử đầu hàng đợi
   * **back()**: Trả về phần tử cuối hàng đợi
   * **empty()**: Kiểm tra hàng đợi trống hay không.
2. **Độ phức tạp: O(1)**
3. **Cài đặt**

- Để sử dụng Queue, chúng ta có thể sử dụng các thư viện hỗ trợ có sẵn của C++ (#include<queue> )

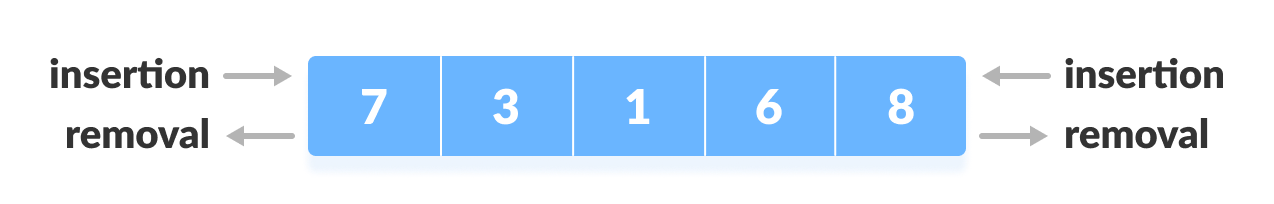
- Cú pháp

queue <object\_type> queue\_name;



1. **DEQUE**
2. **Deque là gì**

* Deque hoặc Double Ended Queue là một loại hàng đợi trong đó việc chèn và loại bỏ các phần tử có thể được thực hiện từ phía trước hoặc phía sau. Do đó, nó không tuân theo quy tắc FIFO (First In First Out)



1. **Các thao tác trên Deque**

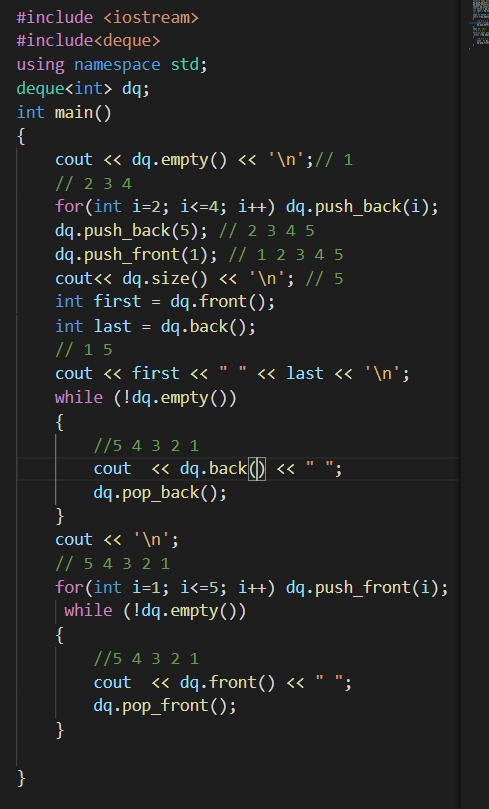
* push\_front(x): Đẩy x vào đầu deque
* push\_back(x): Đẩy x vào cuối deque
* pop\_front(): Loại bỏ phần tử ở đầu deque
* pop\_back(): Loại bỏ phần tử ở cuối deque
* empty(): Kiểm tra Deque có rỗng không
* size(): Trả về số phần tử đang có trong deque

1. **Độ phức tạp**

Độ phức tạp về thời gian của tất cả các hoạt động trên là O(1).

1. **Cài đặt**

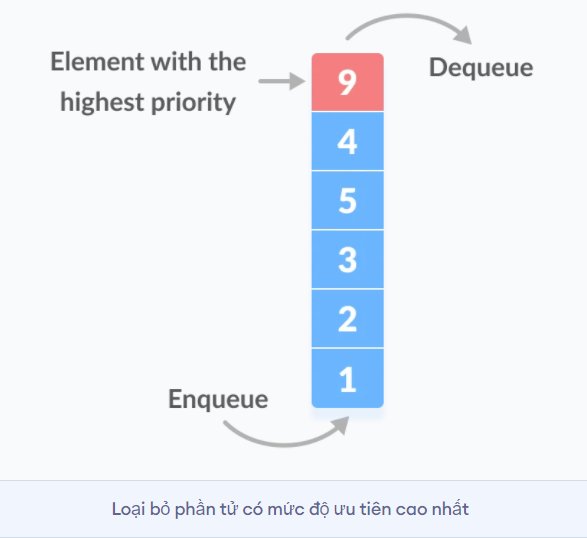
* #include<deque> với cú pháp: deque <object\_type> deque\_name;



1. **PRIORITY QUEUE (hàng đợi ưu tiên)**
2. **Khái niệm**
3. **Priority queue là gì**

* **Hàng đợi ưu tiên** hay **Priority Queue** là một loại hàng đợi đặc biệt, trong đó mỗi phần tử được liên kết với một mức độ ưu tiên và được thực thi theo mức độ ưu tiên của nó.
* Nếu các phần tử có cùng mức độ ưu tiên, chúng sẽ được thực thi theo thứ tự trong hàng đợi. Nói chung, giá trị của phần tử được xem xét để gán mức độ ưu tiên.
* Mục tiêu chính của hàng đợi ưu tiên đó là điều chỉnh để có thể lấy phần tử có độ ưu tiên nhất ra trước. Khi thêm phần tử mới vào thì sắp xếp cho đúng thứ tự ưu tiên lại.

***Ví dụ:*** Phần tử có giá trị cao nhất được coi là phần tử có mức ưu tiên cao nhất. Tuy nhiên, trong các trường hợp khác, chúng ta có thể giả sử phần tử có giá trị thấp nhất là phần tử có mức ưu tiên cao nhất.



1. **Sự khác biệt giữa Priority Queue và Queue thông thường**

* Trong một **queue** thông thường, **quy tắc** **FIFO** được thực hiện, trong khi đó, trong **hàng đợi ưu tiên**, các giá trị được lấy dựa trên cơ sở **mức độ ưu tiên**. Phần tử có mức ưu tiên cao nhất sẽ ra trước.

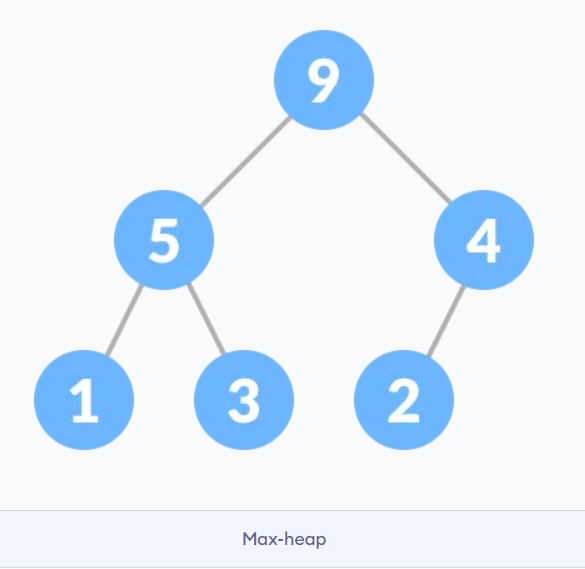
1. **Triển khai hàng đợi ưu tiên**

* Hàng đợi ưu tiên có thể được triển khai bằng cách sử dụng kiểu cấu trúc mảng, danh sách liên kết, cấu trúc dữ liệu heap hoặc cây tìm kiếm nhị phân. Trong số các cấu trúc dữ liệu này, cấu trúc dữ liệu heap cung cấp việc triển khai hiệu quả cho các hàng đợi ưu tiên hơn.

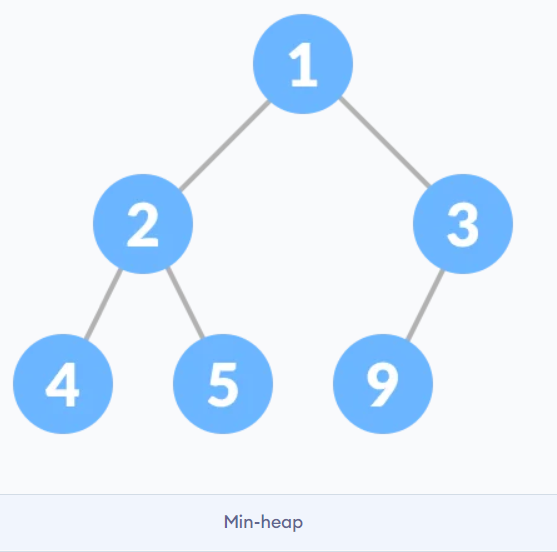
1. **Cấu trúc Heap**

Cấu trúc dữ liệu Heap là một cây nhị phân hoàn chỉnh thỏa mãn thuộc tính heap, trong đó bất kỳ nút đã cho nào đều:

* Luôn lớn hơn các nút con của nó và khóa của nút gốc là lớn nhất trong số tất cả các nút khác. Thuộc tính này còn được gọi là thuộc tính Max heap



* Luôn nhỏ hơn các nút con và khóa của nút gốc là nhỏ nhất trong số tất cả các nút khác. Thuộc tính này còn được gọi là thuộc tính Min heap.



1. **Các hoạt động trên priority queue**

* insert (item, priority): Chèn 1 phần tử vào hàng đợi ưu tiên
* deleteHighestPriority (): Xoá 1 phần tử khỏi hàng đợi ưu tiên
* getHighestPriority (): Trả về phần tử có mức độ ưu tiên cao nhấ

**4. Các thao tác trên pqueue**

**- push():** Chèn một phần tử vào hàng đợi ưu tiên. Phần tử được thêm vào vùng chứa hàng đợi ưu tiên và kích thước của hàng đợi tăng lên 1. Đầu tiên, phần tử được thêm vào phía sau và đồng thời các phần tử của hàng đợi ưu tiên tự sắp xếp lại theo mức độ ưu tiên.

**- pop():** loại bỏ phần tử trên cùng của hàng đợi ưu tiên.

**- empty():** kiểm tra xem vùng chứa hàng đợi ưu tiên có trống hay không.

- **size():** trả về kích thước của vùng chứa hàng đợi ưu tiên hoặc số phần tử trong vùng chứa.

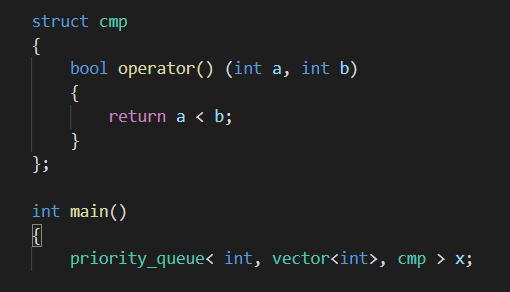
- **top():** tham chiếu đến phần tử trên cùng (hoặc phần tử lớn nhất) của hàng đợi ưu tiên.

**5. Cách sử dụng priority queue trong thư viện STL:**

* Priority queue được thiết kế để phần tử ở đầu luôn luôn lớn nhất (theo quy ước về độ ưu tiên nào đó) so với các phần tử khác.
* Priority queue giống như một heap mà ở đây là heap max, tức là phần tử có độ ưu tiên lớn nhất có thể được lấy ra và các phần tử khác được chèn vào bất kì.
* Để sử dụng ta cần khai báo thư viện queue.
* Phép toán mặc định khi sử dụng priority queue là phép toán less (nhỏ hơn).
* Khai báo: **priority\_queue<int> x;**
* Ngoài ra còn có các phép toán khác:
* greater: lớn hơn

**priority\_queue< int, vector<int>, greater<int> > x;**

* equal\_to: bằng
* not\_equal\_to: không bằng
* greater\_equal: lớn hơn bằng
* less\_equal: nhỏ hơn bằng
* Người dùng cũng có thể tự định nghĩa cách so sánh



1. **BÀI TẬP CHUẨN BỊ**
   * + 1. [**https://ideone.com/p1fv8N**](https://ideone.com/p1fv8N)