


**STACK – QUEUE**  
**PRIORITY QUEUE**

# 1. Cấu trúc dữ liệu ngăn xếp:

## a. Giới thiệu:


 **Ngăn xếp (stack)** là một cấu trúc dữ liệu có **quan hệ mật thiết** với cơ chế hoạt động của đệ quy. Để hiểu được cách hàm đệ quy hoạt động, ta cần **nắm được cách hoạt động** của cấu trúc dữ liệu ngăn xếp,

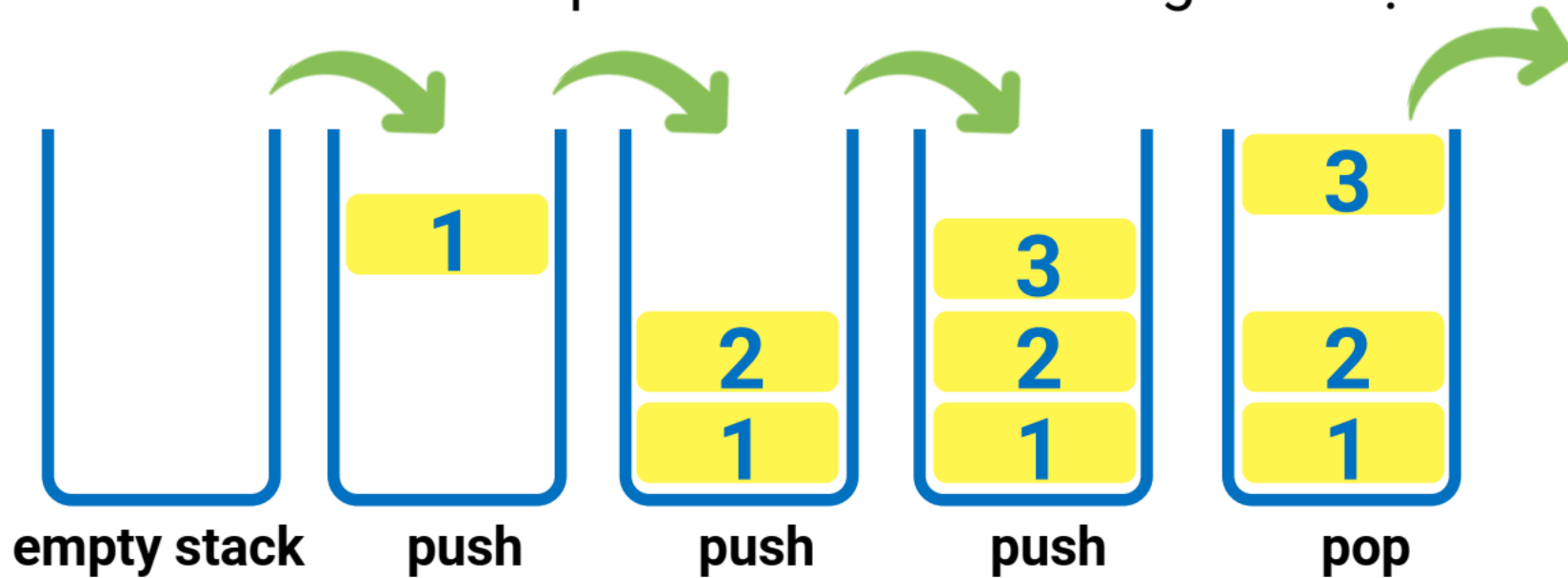



**Ngăn xếp** là một cấu trúc dữ liệu hỗ trợ **2 thao tác push và pop**. Trong đó push giúp thêm 1 phần tử vào đỉnh ngăn xếp, pop giúp xóa 1 phần tử khỏi đỉnh ngăn xếp. Cả 2 thao tác này đều **được thực hiện ở đỉnh ngăn xếp**.

# 1. Cấu trúc dữ liệu ngăn xếp:

## a. Giới thiệu:

 Ngăn xếp hoạt động theo nguyên tắc viết tắt là **LIFO (Last In First Out)** nghĩa là vào cuối thì ra đầu. Các phần tử vào cuối cùng sẽ được ra đầu tiên.



 Trong chương trình tồn tại một bộ nhớ là **bộ nhớ ngăn xếp**, cách hoạt động của bộ nhớ này **tương tự** như cách hoạt động của cấu trúc dữ liệu ngăn xếp.

# 1. Cấu trúc dữ liệu ngăn xếp:

## b. Các hàm phổ biến của stack:

**Khai báo ngăn xếp:** `stack<data_type> stack_name;`

**Ví dụ:** `stack<int> st;`

Hàm	Chức năng
<b>push(x)</b>	Thêm x vào đỉnh ngăn xếp
<b>pop()</b>	Xóa phần tử khỏi ngăn xếp (chú ý nếu ngăn xếp rỗng việc pop sẽ gây lỗi)
<b>size()</b>	Trả về số lượng phần tử trong ngăn xếp
<b>empty()</b>	Kiểm tra ngăn xếp rỗng
<b>top()</b>	Lấy về phần tử ở đỉnh ngăn xếp nhưng không xóa

## 2. Cấu trúc dữ liệu hàng đợi:

### a. Giới thiệu:



**Hàng đợi (Queue)** là một cấu trúc dữ liệu quan trọng với cách thức hoạt động là **FIFO : First In First Out**, tức là phần tử nào vào trước sẽ được ra trước.

- **Thao tác enqueue:** Thêm 1 phần tử vào cuối hàng đợi
- **Thao tác dequeue:** Xóa 1 phần tử khỏi đầu hàng đợi



## 2. Cấu trúc dữ liệu hàng đợi:

### b. Các hàm phổ biến của hàng đợi:

**Khai báo hàng đợi:** `queue<data_type> queue_name;`

**Ví dụ:** `queue<int> q;`

Hàm	Chức năng
<b>push(x)</b>	Thêm x vào cuối hàng đợi
<b>front()</b>	Truy cập vào phần tử ở đầu hàng đợi
<b>pop()</b>	Xóa phần tử khỏi đầu hàng đợi
<b>size()</b>	Trả về số phần tử trong hàng đợi
<b>empty()</b>	Kiểm tra hàng đợi rỗng
<b>back()</b>	Truy cập vào phần tử ở cuối hàng đợi

### 3. Hàng đợi ưu tiên:



**Hàng đợi ưu tiên (priority\_queue)** khác với hàng đợi thông thường đó là nó không phụ thuộc vào thứ tự bạn thêm phần tử vào hàng đợi mà phần tử ở đỉnh hàng đợi luôn là lớn nhất hoặc nhỏ nhất. Priority\_queue trong C++ STL mặc định thì phần tử ở đầu hàng đợi luôn là lớn nhất.

#### Ví dụ hàng đợi mặc định:

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
int main(){
    priority_queue<int> Q;
    Q.push(3); Q.push(1);
    Q.push(2); Q.push(4);
    cout << Q.top() << endl; // 4
    Q.pop(); // xoa 4
    Q.push(0); Q.push(5);
    cout << Q.top() << endl;
}
```

**OUTPUT**

4 5

### 3. Hàng đợi ưu tiên:

Hàng đợi mà phần tử ở đỉnh là nhỏ nhất

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

using ll = long long;

int main(){
    priority_queue<int, vector<int>,greater<int>> Q;
    Q.push(3);
    Q.push(1);
    Q.push(2);
    Q.push(4);
    cout << Q.top() << endl; // 1
    Q.pop(); // xoa 1
    Q.push(0);
    Q.push(5);
    cout << Q.top() << endl; // 0
}
```

**OUTPUT**

1 0