Big-O Blue

LECTURE 04

STACK & QUEUE

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

STACK
(NGĂN XẾP)

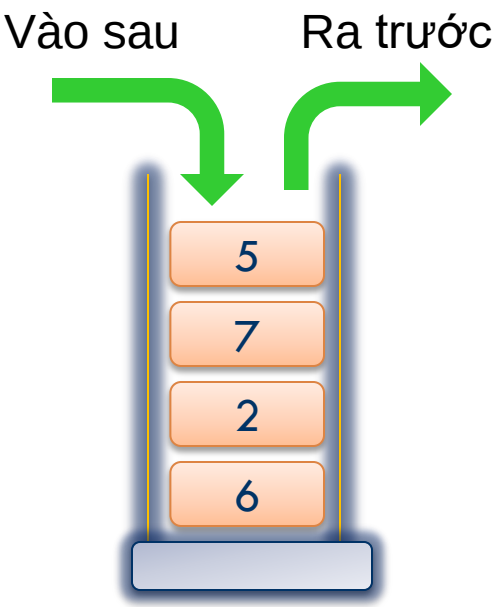
Stack là gì?

Stack (ngăn xếp) là dãy phần tử hoạt động theo cơ chế LIFO (**L**ast **I**n **F**irst **O**ut) vào sau ra trước, giống như ngăn tủ xếp đồ.

C++: stack.

Python: list = [ ].

Java: stack.



Cách khai báo và sử dụng

C++

Thư viện:

```
#include <stack>
using namespace std;
```

Khai báo:

```
stack<data_type> variable_name;
```

Ví dụ:

```
stack<int> s;
```

python

Trong Python không có cấu trúc stack riêng, sử dụng list để biểu diễn stack.

Khai báo:

```
variable_name = []
```

Ví dụ:

```
s = []
```

Cách khai báo và sử dụng

Thư viện:

```
import java.util.Stack
```

Khai báo:

```
Stack<data_type> variable_name = new Stack<data_type>();
```

Ví dụ:

```
Stack<Integer> s = new Stack<Integer>();
```

Kiểm tra stack rỗng

C++

python

empty()

```
stack<int> s;
if(s.empty()== true)
    cout<<"stack is empty!";
else
    cout<<"stack is not empty!";
```

len()

```
s = []
if len(s) == 0:
    print("stack is empty!")
else:
    print("stack is not empty!")
```

Kết quả

stack is empty!

Kiểm tra stack rỗng

empty/isEmpty: Kiểm tra xem stack có rỗng hay không.

```
Stack<Integer> S = new Stack<Integer>();
if (S.empty())
    System.out.print("stack is empty!");
else
    System.out.print("stack is not empty!");
```

Kết quả

stack is empty!

Thêm phần tử vào stack

C++

python

push(value)

```
stack<int> s;
s.push(5);
s.push(7);
s.push(3);
```

append(obj)

```
s = []
s.append(5)
s.append(7)
s.append(3)
```

Kết quả

Thêm phần tử vào stack

push(obj)/add(obj): Thêm một phần tử vào trong stack.


```
Stack<Integer> s = new Stack<Integer>();
s.push(5);
s.add(7);
s.add(3);
```

Kết quả

Xóa phần tử khỏi stack

C++

pop()

```
s.pop();
```

python

pop()

```
s.pop();
```

Kết quả

Xóa phần tử khỏi stack

pop(): Xóa và trả về giá trị phần tử cuối cùng ra khỏi stack.

```
s.pop();
```

Kết quả

Lấy giá trị trên cùng stack

C++

top(): Lấy giá trị phần tử ở trên cùng stack.

```
int value = s.top();
cout<<value;
```

python

Lấy giá trị phần tử ở trên cùng stack bằng cách, dùng [ ] để lấy ra phần tử ở vị trí cuối stack.

```
value = s[-1]
print(value)
```

Kết quả

3

Lấy giá trị trên cùng stack

peek(): Lấy giá trị phần tử ở trên cùng stack, không xóa phần tử đó.

```
int value = s.peek();
System.out.print(value);
```

Kết quả

3

Lấy kích thước của stack

C++

size()

```
int n = s.size();
cout<<n;
```

python

len(obj)

```
n = len(s)
print(n)
```

Kết quả

5

Lấy kích thước của stack

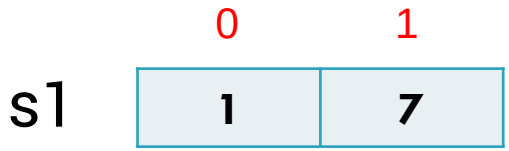
size(): Lấy kích thước của stack.

```
int n = s.size();
System.out.print(n);
```

Kết quả

5

Hóa đổi 2 stack với nhau

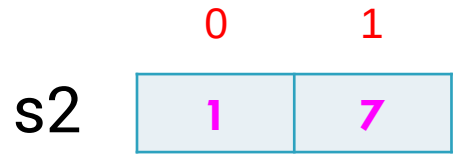


swap(other stack)

```
s1.swap(s2);
```

Hoán đổi 2 stack với nhau: Python không hỗ trợ hàm swap, tuy nhiên có thể sử dụng phép gán để swap.

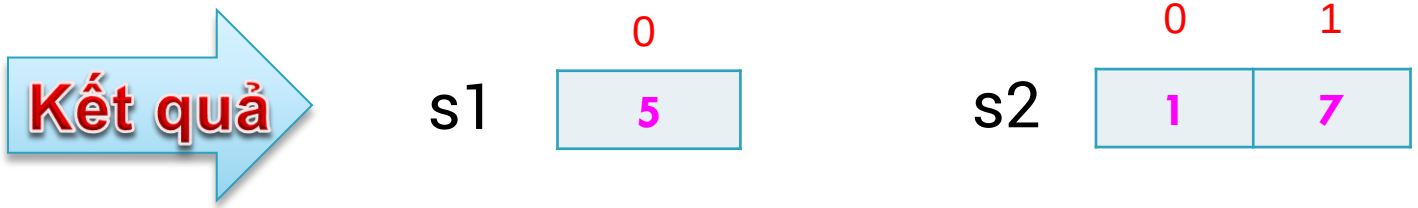
```
s1,s2 = s2,s1
```



Hóa đổi 2 stack với nhau

Java không hỗ trợ hàm swap. Chỉ có thể tự viết lệnh swap 2 stack tương tự như swap 2 biến primitive type.

```
Stack<Integer> temp = s;
s = t;
t = temp;
```



QUEUE
(HÀNG ĐỢI)

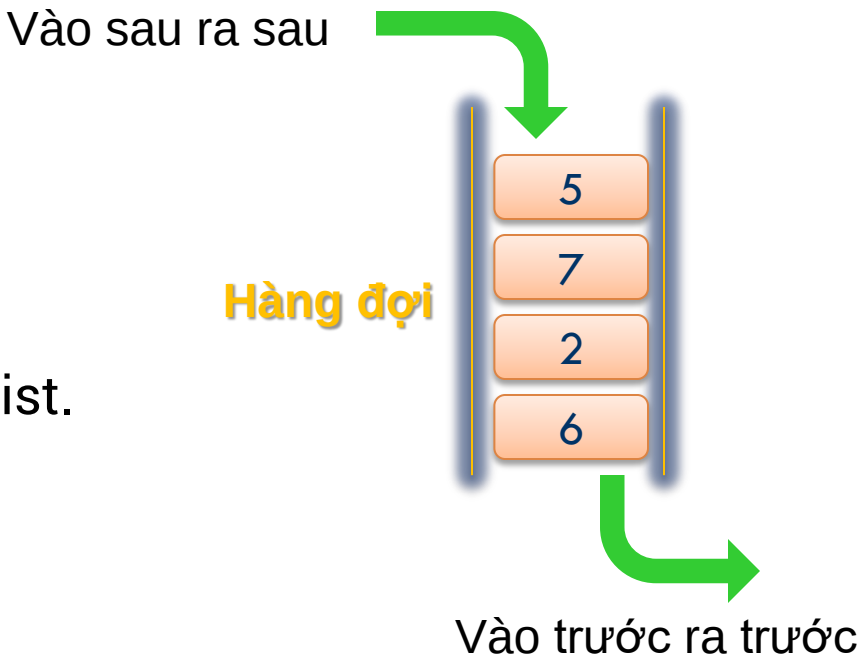
Queue là gì?

Định nghĩa: Queue (hàng đợi) là dãy phần tử hoạt động theo cơ chế FIFO (First In First Out) vào trước ra trước, giống như việc xếp hàng mua vé.

C++: queue.

Python: queue

Java: Queue new LinkedList.



Cách khai báo và sử dụng

C++

Thư viện:

```
#include <queue>
using namespace std;
```

Khai báo:

```
queue<data_type> variable_name;
```

Ví dụ:

```
queue<int> q;
```

python

Thư viện:

```
import queue
import Queue
```

Khai báo:

```
 variable_name = queue.Queue()
(Py 3.x)
 variable_name = Queue.Queue()
(Py 2.x)
```

Ví dụ:

```
q = queue.Queue() # Py 3.x
q = Queue.Queue() # Py 2.x
```

Cách khai báo và sử dụng

Queue trong Java chỉ là 1 interface, nên không thể dùng trực tiếp mà thường được implement bằng LinkedList.

Thư viện:

```
import java.util.LinkedList
```

Khai báo:

```
Queue<data_type> queue = new LinkedList<data_type>();
```

Ví dụ:

```
Queue<Integer> queue = new LinkedList<Integer>();
```

Kiểm tra queue rỗng không

C++

empty()

```
queue<int> q;
if(q.empty()== true)
    cout<<"queue is empty!";
else
    cout<<"queue is not empty!";
```

python

empty()

```
q = queue.Queue()
if q.empty():
    print("queue is empty!")
else:
    print("queue is not empty!")
```

Kết quả

queue is empty!

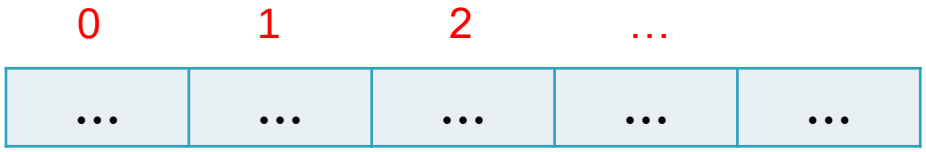
Kiểm tra queue rỗng không

isEmpty(): Kiểm tra xem queue có rỗng hay không?

```
Queue<Integer> queue = new LinkedList<Integer>();
if (queue.isEmpty())
    System.out.print("queue is empty!")
else:
    System.out.print("queue is not empty!")
```

queue is empty!

Thêm phần tử vào queue



C++

python

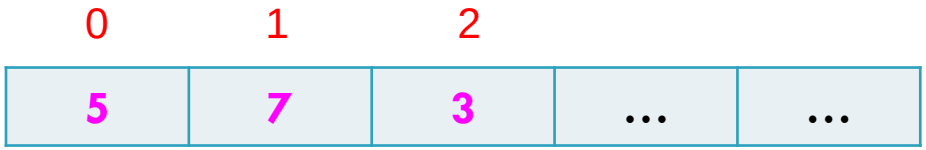
push(value)

```
queue<int> q;
q.push(5);
q.push(7);
q.push(3);
```

put(obj)

```
q = queue.Queue()
q.put(5)
q.put(7)
q.put(3)
```

Kết quả

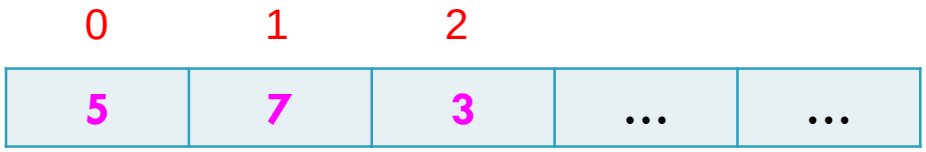


Thêm phần tử vào queue

add(obj): Thêm một phần tử vào trong queue.

```
Queue<Integer> queue = new LinkedList<Integer>();
queue.add(5);
queue.add(7);
queue.add(3);
```

Kết quả



Xóa phần tử khỏi queue

C++

pop()

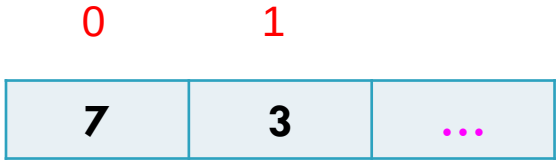
```
q.pop();
```

python

get(): Lấy và xóa

```
q.get();
```

Kết quả



Xóa phần tử khỏi queue

remove(): Lấy giá trị và xóa một phần tử đầu tiên ra khỏi queue.

```
queue.remove();
```

Kết quả

Lấy phần tử trên đầu hàng đợi

C++

front()

```
int value = q.front();
cout<<value;
```

python

queue[0]

```
value = q.queue[0]
print(value)
```

Kết quả

5

Lấy phần tử trên đầu hàng đợi

peek(): Lấy giá trị phần tử ở trên cùng queue, không xóa phần tử đó.

```
int value = s.peek();
System.out.println(value);
```

5

Lấy kích thước của queue

C++

size()

```
int n = q.size();
cout<<n;
```

python

qsize()

```
n = q.qsize()
print(n)
```

Kết quả

5

Lấy kích thước của queue

0	1	2	3	4
5	7	8	3	6

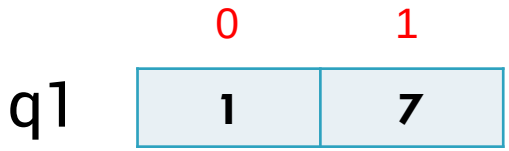
size(): Trả về kích thước của queue.

```
int n = queue.size()
System.out.print(n);
```

Kết quả

```
5
```

Hóa đổi 2 queue với nhau



C++

swap(other queue)

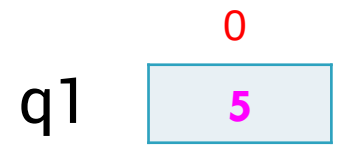
```
q1.swap(q2);
```

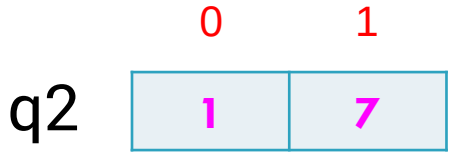
python

Hoán đổi 2 queue với nhau: Python không hỗ trợ hàm swap, tuy nhiên có thể sử dụng phép gán để swap.

```
q1,q2 = q2,q1
```

Kết quả

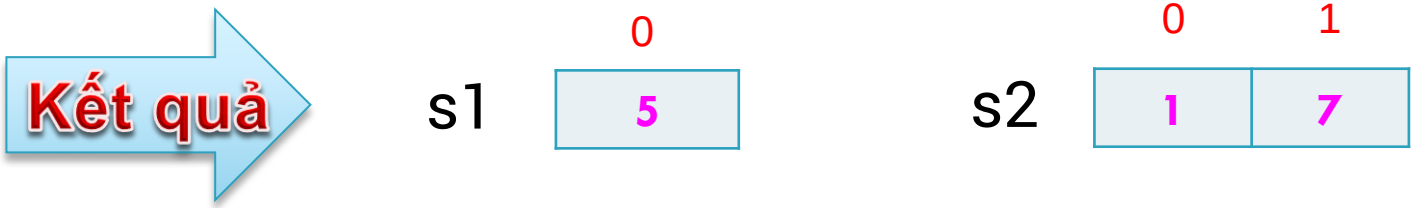




Hoán đổi 2 queue với nhau

Java không hỗ trợ hàm swap, sử dụng phương pháp như stack, viết hàm swap cho 2 queue với nhau.

```
Queue<Integer> temp = s;
s = t;
t = temp;
```



Hỏi đáp