Big-O Blue

LECTURE 04

STACK & QUEUE

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

STACK
(NGĂN XẾP)

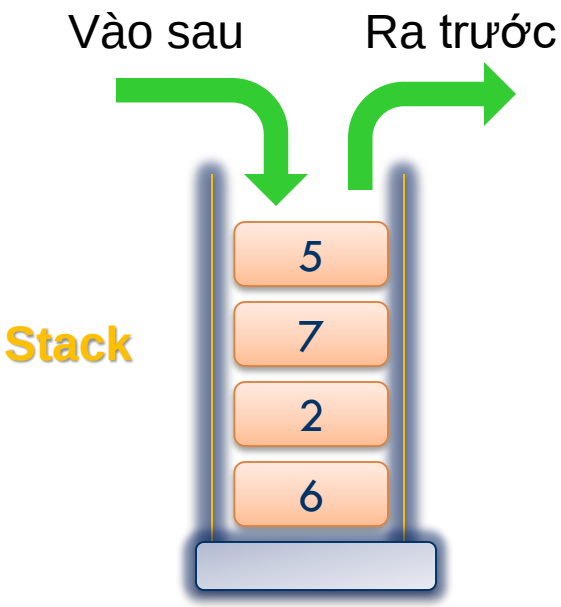
Stack

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, có thể hình dung stack giống như ngăn tủ xếp đồ trong thực tế.

C++: **stack**

Python: **list = [ ]**

Java: **Stack**



Cách khai báo và sử dụng

Thư viện:

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

Khai báo:

```
stack<data_type> variable;
```

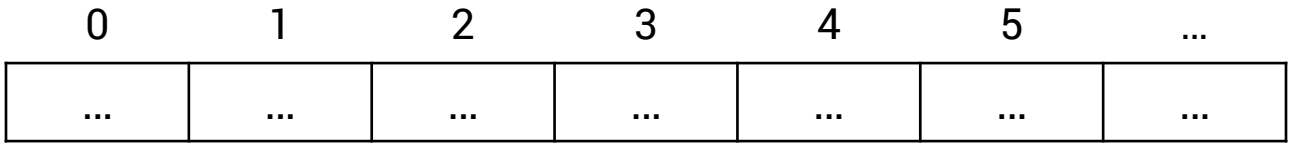
```
stack<int> s;
```

Trong Python có thể sử dụng list để biểu diễn stack.

Khai báo:

```
variable = []
```

```
s = []
```



Cách khai báo và sử dụng

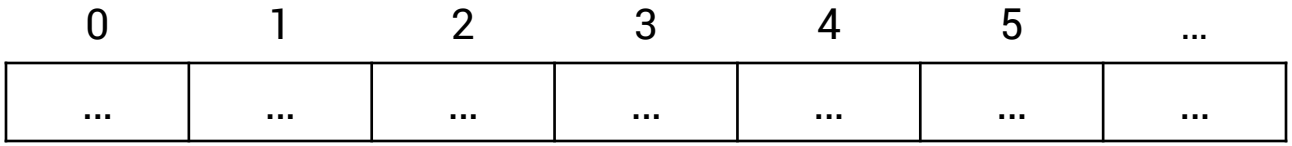
Thư viện:

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

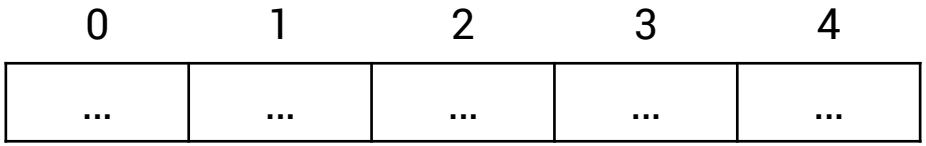
Khai báo:

```
Stack<E> variable = new Stack<E>();
```

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



Thêm phần tử vào stack



C++

push(value)

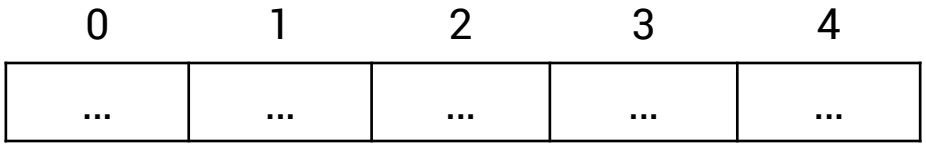
```
stack<int> s;
s.push(5);
s.push(7);
s.push(3);
```

python

append(obj)

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

Thêm phần tử vào stack



push(E)

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

add(E): là một hàm tương tự như push(E)

Xóa phần tử trên cùng của stack

0	1	2
5	7	3

pop(): hàm này chỉ xóa không lấy được giá trị trả về.

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

0	1	...
5	7	...

pop(): hàm này ngoài xóa ra thì có thể lấy được giá trị trả về.

```
value = s.pop();
print(value)
```

0	1	...
5	7	...

3

Xóa phần tử trên cùng của stack

0	1	2
5	7	3

pop(): hàm này ngoài xóa ra thì có thể lấy được giá trị trả về.

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

0	1	...
5	7	...

3

Lấy giá trị trên cùng stack

0	1	2
5	7	3

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

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

Lấy giá trị phần tử ở trên cùng stack bằng cách dùng [ ].

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

3

Lấy giá trị trên cùng stack

0	1	2
5	7	3

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

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

3

Lấy kích thước của stack

0	1	2
5	7	3

size()

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

len(obj)

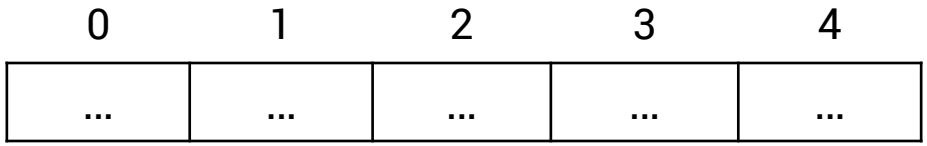
```python
n = len(s)
print(n)
```

size()


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

3

Kiểm tra stack rỗng



C++

empty()

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

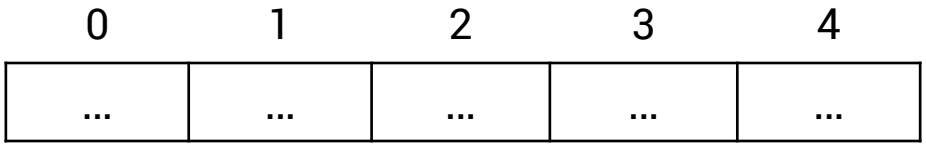
python

len()

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

stack is empty!

Kiểm tra stack rỗng



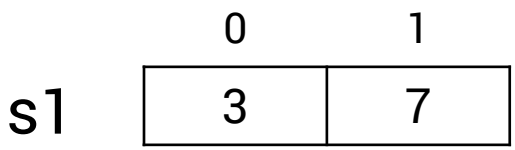
empty()

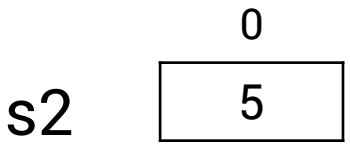
```
Stack<Integer> s = new Stack<Integer>();
if (s.empty())
    System.out.print("stack is empty!");
else
    System.out.print("stack is not empty!");
```

stack is empty!

isEmpty(): là một hàm tương tự như empty()

Hoán đổi 2 stack với nhau





C++

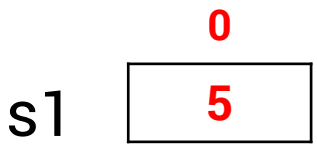
swap(stack &other)

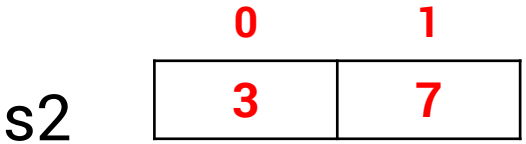
```
s1.swap(s2);
```

python

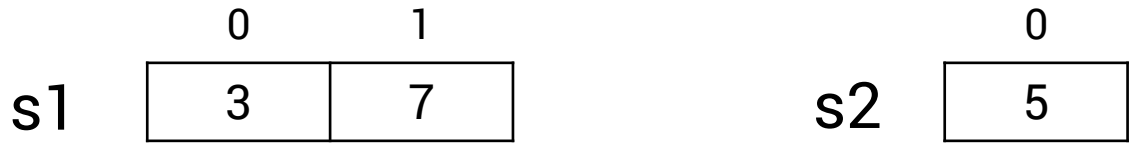
Python không hỗ trợ hàm swap, tuy nhiên có thể sử dụng phép gán để swap.

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





Hoán 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 = s1;
s1 = s2;
s2 = temp;
```

QUEUE
(HÀNG ĐỢI)

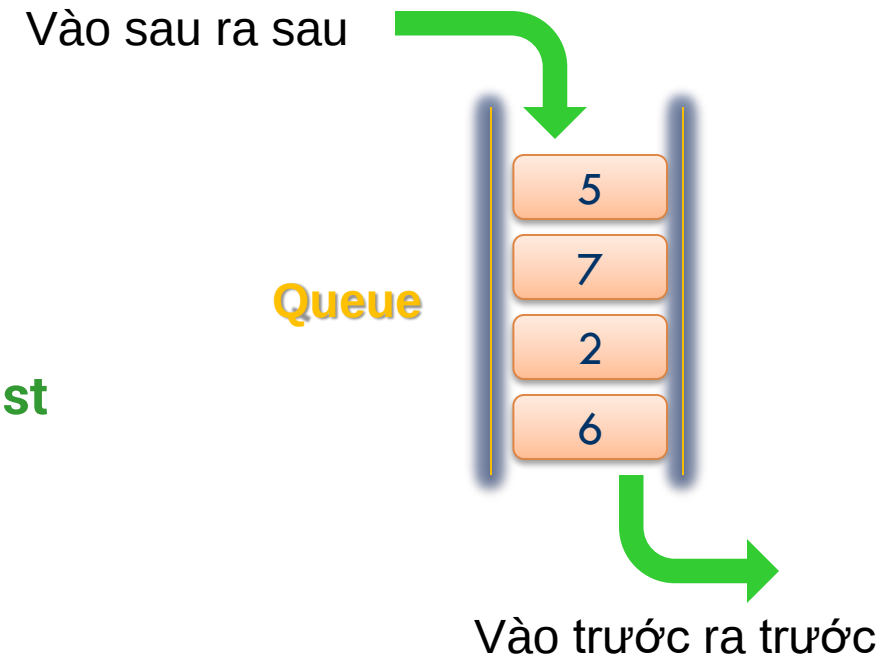
Queue

Queue (Hàng đợi): là dãy phần tử hoạt động theo cơ chế FIFO (First In First Out) vào trước ra trước, có thể hình dung queue giống như hàng đợi xếp hàng mua vé trong thực tế.

C++: **queue**

Python: **queue**

Java: **Queue new LinkedList**



Cách khai báo và sử dụng

Thư viện:

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

Khai báo:

```
queue<data_type> variable;
```

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

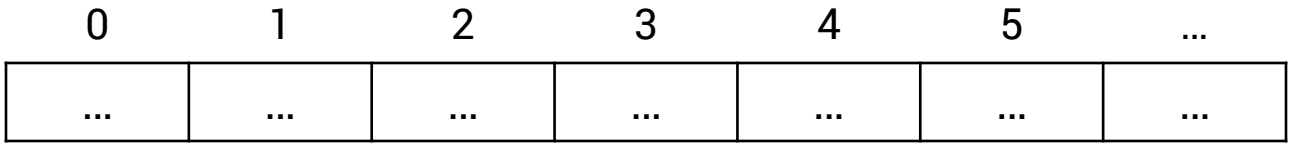
Thư viện:

```
import queue
import Queue
```

Khai báo:

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

```
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 cài đặt bằng LinkedList.

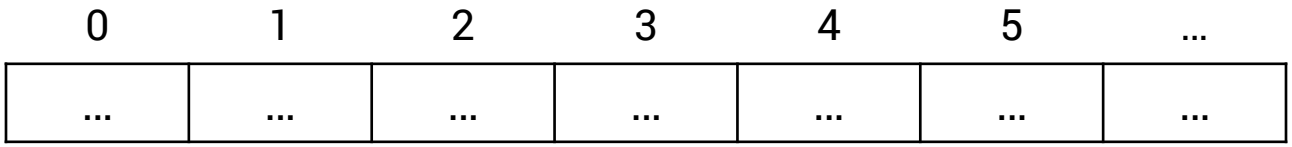
Thư viện:

```
import java.util.LinkedList;
import java.util.Queue;
```

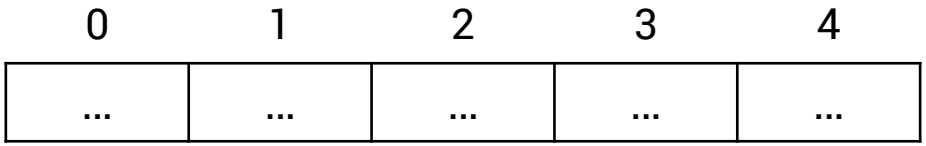
Khai báo:

```
Queue<E> variable = new LinkedList<E>();
```

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



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)
```

Thêm phần tử vào queue

0	1	2	3	4
...	...	...	...	...

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

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

0	1	2	...	...
5	7	3	...	...

offer(E): là một hàm tương tự như add(E)

Xóa phần tử khỏi queue

0	1	2
5	7	3

pop(): hàm này chỉ xóa không lấy được giá trị trả về.

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

0	1	...
7	3	...

get(): hàm này ngoài xóa ra thì có thể lấy được giá trị trả về.

```
value = s.get()
print(value)
```

0	1	...
7	3	...

5

Xóa phần tử khỏi queue

0	1	2
5	7	3

remove(): hàm này ngoài xóa ra thì có thể lấy được giá trị trả về.

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

0	1	…
7	3	…

5

poll(): là một hàm tương tự như remove()

Lấy phần tử đầu queue

0	1	2
5	7	3

C++

front()

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

python

queue[0]

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

5

Lấy phần tử đầu queue

0	1	2
5	7	3

peek()

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

5

element(): là một hàm tương tự như peek()

Lấy kích thước của queue

0	1	2
5	7	3

size()

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

qsize()

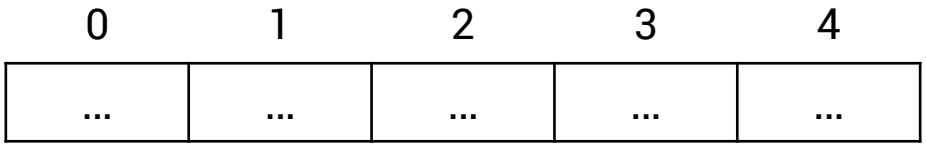
```python
n = q.qsize()
print(n)
```

size()

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

3

Kiểm tra queue rỗng



C++

empty()

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

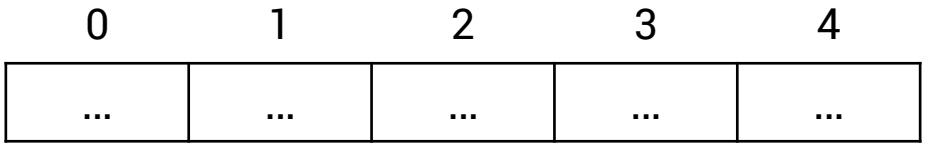
python™

empty()

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

queue is empty!

Kiểm tra queue rỗng

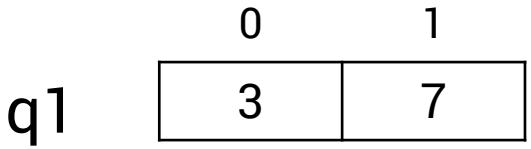


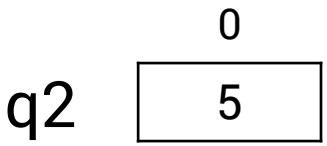
isEmpty()

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

queue is empty!

Hoán đổi 2 queue với nhau





C++

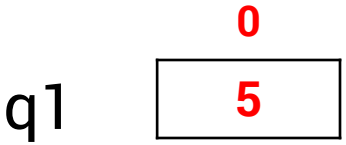
swap(other& queue)

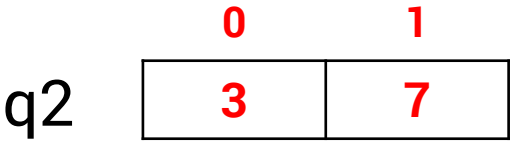
```
q1.swap(q2);
```

python

Python không hỗ trợ hàm swap, tuy nhiên có thể sử dụng phép gán để swap.

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





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 = q1;
q1 = q2;
q2 = temp;
```

Bài toán minh họa

SPOJ.com **Transform the Expression**

Cho một danh sách các biểu thức toán học chỉ chứa các phép toán +, -, *, /, ^, các ký tự a, b, …, z và dấu ngoặc ( ) thể hiện sự ưu tiên. Hãy chuyển các biểu thức toán học này về dạng biểu diễn Ký pháp Ba Lan Ngược (Reverse Polish Notation).

Input:

Dòng đầu tiên chứa một số nguyên *t* (t ≤ 100) – số lượng biểu thức cần biến đổi.

t dòng tiếp theo, mỗi dòng chứa một biểu thức có độ dài không quá 400 ký tự.

Output:

Với mỗi biểu thức được cho, in ra trên một dòng là dạng biến đổi Ký pháp Ba Lan ngược tương ứng.

Bài toán minh họa

Ví dụ:

3 (a+(b*c)) ((a+b)*(z+x)) ((a+t)*((b+(a+c))^(c+d)))	abc*+ ab+zx+* at+bac++cd+^*

Hướng dẫn giải

- **Bước 1:** Đọc lần lượt từng bộ test vào biến chuỗi để xử lý.

```
((a+b)*(z+x))
```

O(len(expression))

	0	1	2	3	4	5	6	7	8	9	10	11	12
s	(	(	a	+	b	)	*	(	z	+	x	)	)

- **Bước 2:** Duyệt qua chuỗi cần xử lý từ 0 đến s.size() – 1. Thực hiện thao thác như sau:

  Phân tích đáp án của bài toán ta thấy. `ab+zx+*`

  - Dấu ngoặc không được in ra mà để xử lý tính toán ưu tiên biểu thức nào.

    O(len(expression))

  - Ưu tiên in các biến khi đã có đủ các ngoặc rồi mới in phép toán.

Hướng dẫn giải

	0	1	2	3	4	5	6	7	8	9	10	11	12
s	(	(	a	+	b	)	*	(	z	+	x	)	)

STT	Ký tự	Thực hiện	stack	Kết quả
0	(	Bỏ qua	...	...
1	(	Bỏ qua	...	...
2	a	In ra kết quả	...	a
3	+	Push vào stack	+	a
4	b	In ra kết quả	+	ab
5	)	In kết quả trong stack	...	ab+
6	*	Bỏ vào stack	*	ab+
7	(	Bỏ qua	*	ab+
8	z	In ra kết quả	*	ab+z

Hướng dẫn giải

	0	1	2	3	4	5	6	7	8	9	10	11	12
s	(	(	a	+	b	)	*	(	z	+	x	)	)

STT	Ký tự	Thực hiện	stack	Kết quả
8	z	In ra kết quả	*	ab+z
9	+	Bỏ vào stack	*+	ab+z
10	x	In ra kết quả	*+	ab+zx
11	)	In kết quả trong stack	*	ab+zx+
12	)	In kết quả trong stack	...	ab+zx+*

Đáp án: `ab+zx+*`

Time Complexity: O(t * len(expression))

Source Code Transform the Expression

```
1.  #include <iostream>
2.  #include <stack>
3.  #include <string>
4.  using namespace std;
5.  void transform(string expression)
6.  {
7.      stack<char> s;
8.      for (char symbol : expression)
9.      {
10.         if (isalpha(symbol))
11.             cout << symbol;
12.         else if (symbol == ')')
13.         {
14.             cout << s.top();
15.             s.pop();
16.         }
17.         else if (symbol != '(')
18.             s.push(symbol);
19.     }
20.     cout << endl;
21. }
```

Source Code Transform the Expression

```
22. int main()
23. {
24.     int t;
25.     string expression;
26.     cin >> t;
27.     for (int i = 0; i < t; i++)
28.     {
29.         cin >> expression;
30.         transform(expression);
31.     }
32.     return 0;
33. }
```

Source Code Transform the Expression

```
1.  def transform(expression):
2.      s = []
3.      for symbol in expression:
4.          if symbol.isalpha():
5.              print(symbol, end='')
6.          elif symbol == ')':
7.              print(s.pop(), end='')
8.          elif symbol != '(':
9.              s.append(symbol)
10.     print()
11.
12. t = int(input())
13. for i in range(t):
14.     expression = input()
15.     transform(expression)
```

Source Code Transform the Expression

```
1.  import java.util.Scanner;
2.  import java.util.Stack;
3.  public class Main {
4.      private static void transform(String expression) {
5.          Stack<Character> s = new Stack<>();
6.          for (int i = 0; i < expression.length(); i++) {
7.              char symbol = expression.charAt(i);
8.              if (Character.isLetter(symbol)) {
9.                  System.out.print(symbol);
10.             }
11.             else if (symbol == ')') {
12.                 System.out.print(s.pop());
13.             }
14.             else if (symbol != '(') {
15.                 s.push(symbol);
16.             }
17.         }
18.         System.out.println();
19.     }
```

Source Code Transform the Expression

```
20.     public static void main(String[] args) {
21.         Scanner sc = new Scanner(System.in);
22.         int t = sc.nextInt();
23.         for (int i = 0; i < t; i++) {
24.             String expression = sc.next();
25.             transform(expression);
26.         }
27.     }
28. }
```

Hỏi đáp