

Estructura de datos y algoritmos

Rodrigo Alvarez

rodrigo.alvarez2@mail.udp.cl

Stack

The screenshot shows the VISUALGO.NET website interface for the Stack data structure. The top navigation bar includes the logo, a language dropdown set to 'en', and links for /list, LL, STACK, QUEUE, DLL, and DEQUE. On the right, there is a 'LOGIN' button and a link to 'Exploration Mode'. The main content area is a large black rectangle. On the left side of this area, there is a vertical menu with a left-pointing arrow and four options: 'Create(A)', 'Peek', 'Push', and 'Pop'. At the bottom left of the main area, there is a '1x' zoom indicator. At the bottom right, there is a right-pointing arrow. The footer contains links for 'About', 'Team', 'Terms of use', and 'Privacy Policy'.

```


class Stack {
    class Node {
        int data;
        Node next;
        public Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    Node head;
    public Stack() { this.head = null; }
    public void push(int e) {
        Node newNode = new Node(e);
        if (head != null) {
            newNode.next = head;
        }
        head = newNode;
    }
    public int pop() {
        int e = head.data;
        head = head.next;
        return e;
    }
    public int peek() { return head.data; }
}

```

Complejidad:

- `push` : $O(1)$
- `pop` : $O(1)$
- `peek` : $O(1)$

Queue

 VISUALGO.NET /

en

 /list

LLSTACK**QUEUE**DLLDEQUE

Exploration Mode

LOGIN

<

Create(A)
Peek
Enqueue
Dequeue

■ 1x

AboutTeamTerms of usePrivacy Policy

```
class Queue {
    class Node {
        int data;
        Node next;
        public Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    Node head, tail;
    public Queue() {
        this.head = this.tail = null;
    }
    public void enqueue(int e) {
        Node newNode = new Node(e);
        if (head == null) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
        }
    }
    public int dequeue() {
        int e = head.data;
        head = head.next;
        return e;
    }
    public int peek() {
        return head.data;
    }
}
```

Complejidad:

- enqueue : $O(1)$
- dequeue : $O(1)$
- peek : $O(1)$

- Stack using two queues
- Balanced brackets
- Reverse a doubly linked list
- Add two numbers
- Merge two sorted lists