# 04-A

## 栈与队列

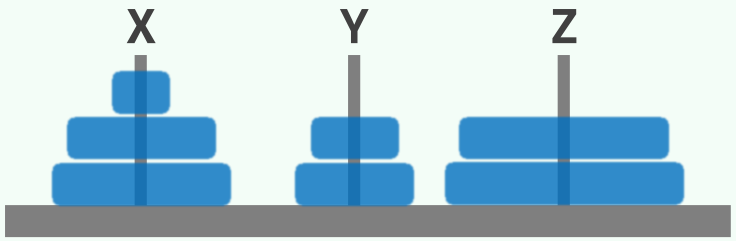
## 栈接口与实现

陛下用群臣，如积薪耳，后来者居上

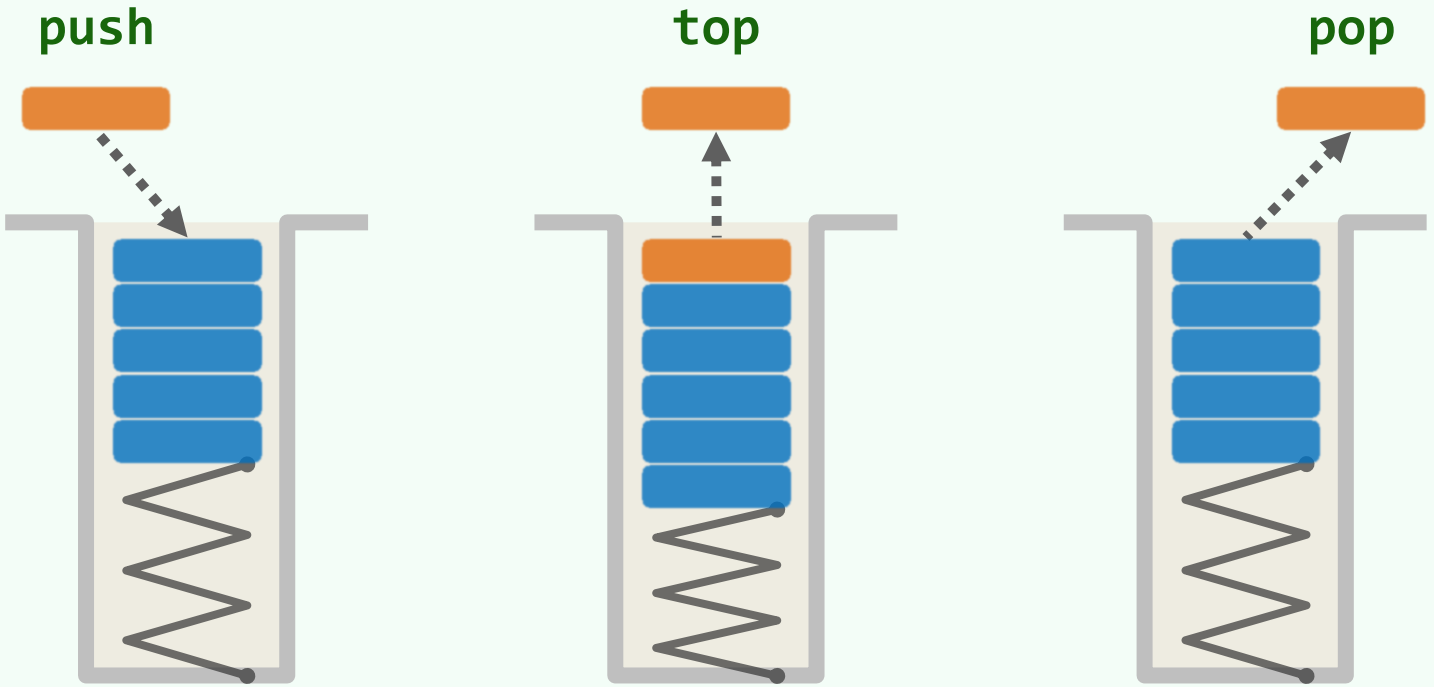
邓俊辉
deng@tsinghua.edu.cn

# 操作与接口

- ❖ 栈（stack）是受限的序列
  - 只能在栈顶（top）插入和删除
  - 栈底（bottom）为盲端




- ❖ 基本接口
  - size() / empty()
  - push() 入栈
  - pop() 出栈
  - top() 查顶




- ❖ 后进先出（LIFO）

  先进后出（FILO）

# 实例

| 操作 | [-- 栈 --x | 输出 |
|---|---|---|
| Stack() | | |
| empty() | | true |
| push(E) | E | |
| push(C) | E C | |
| pop() | E | C |
| push(G) | E G | |
| push(C) | E G C | |
| top() | E G C | C |
| empty() | E G C | false |

| 操作 | [-- 栈 --x | 输出 |
|---|---|---|
| push(K) | E G C K | |
| size() | E G C K | 4 |
| push(F) | E G C K F | |
| empty() | E G C K F | false |
| push(G) | E G C K F G | |
| pop() | E G C K F | G |
| pop() | E G C K | F |
| top() | E G C K | K |
| size() | E G C K | 4 |

# 实现：既然属于线性序列，故可直接基于向量或列表派生

```
template <typename T> class Stack: public Vector<T> { //原有接口一概沿用
public:
   void push( T const & e ) { insert( e ); } //入栈
   T pop() { return remove( size() - 1 ); } //出栈
   T & top() { return (*this)[ size() - 1 ]; } //取顶
}; //以向量首/末端为栈底/顶——颠倒过来呢?
```

- 确认：如此实现的栈接口，均只需 $\mathcal{O}(1)$ 时间
- 课后：基于列表，派生出栈模板类——如此实现的各接口，效率如何?