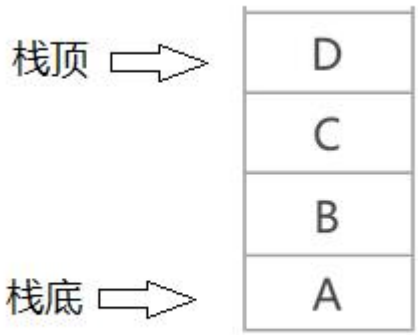
栈（FILO）

概论

基本概念

只能从线性表的一端进行插入和删除元素操作的数据结构即为栈。



应用场景

根据概念我们就可以得出，凡是只在数据结构一端进行插入删除的，并且不需要支持随机存取的场景就可以采用栈结构。

下面列举一些栈的使用场景：

- C / C++ 函数调用是借助 **栈帧** 结构实现的，栈帧即栈结构（有兴趣的读者可以看看这个具体是怎么实现的）
- 编译过程中的语法检查（同上）
- ……

分类

按照代码实现，可以分为两类：**顺序栈** 和 **链式栈**。前者是基于顺序表实现的，后者是基于链表实现的。

方法

> 以下方法用链式栈实现，可以实现动态增长

定义节点

```
typedef int DataType;
typedef struct StackNode{
    DataType _elem;
    struct StackNode* _next;
} Stack;
```

初始化

```
void StackInit(Stack* stack){
    assert(NULL);
    *stack = NULL;
}
```

压栈

```
void Push(Stack *stack, DataType elem){
    assert(stack);
    Stack *tmp = (Stack*)malloc(sizeof(Stack));
    tmp->_elem = elem;
    tmp->_next = stack;
    stack = tmp;
}
```

出栈

```
void Pop(Stack *stack){
    assert(stack);
    if(NULL == *stack){
        return;
    }
    Stack *tmp = stack;
    stack = stack->_next;
    free(tmp);
}
```

获取栈顶

```
Stack* Top(Stack* stack){
    assert(stack);
    return stack; //如果栈为空，则返回NULL
}
```