

# 线性结构

计算机程序设计 -- 数据结构

## 2.栈



# 栈

栈 (stack) 是一种特殊的线性结构, 它**只能在一端进行插入和删除操作**。

能插入和删除的一端**栈顶** (top), 另一端称为**栈底** (bottom)。

不含任何元素的栈称为**空栈**。

只允许在栈顶进行插入和删除, 所以栈的操作是按“**后进先出**” (Last In First Out) 的原则进行的。

栈底 (bottom)

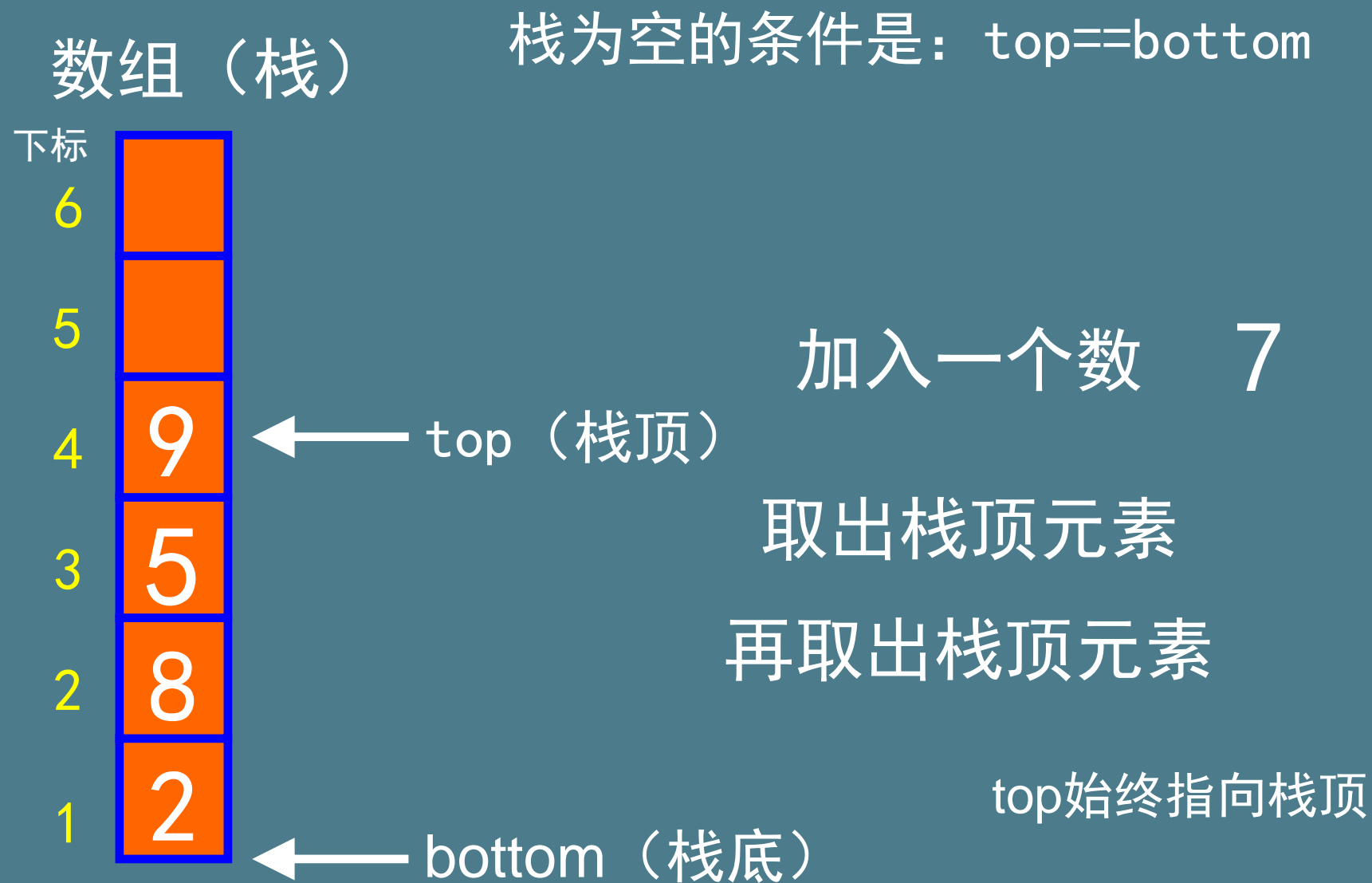
# 栈的实例1：汉诺塔



## 栈的实例2：弹夹



# 用数组模拟栈的“后进先出”



```
#define maxn 100
int Stack[maxn+1];
int top;
```

//插入(压栈)

```
void push(int x)
{
    if(top==maxn) cout<<"full";
    else
    {
        top++;
        Stack[top]=x;
    }
}
```

//删除(弹栈)

```
void pop()
{
    if(top==0) cout<<"empty";
    else
    {
        cout<<Stack[top];
        top--;
    }
}
```

## 考题（初赛）

若S是一个大小为4的栈，若元素1，2，3，4，5，6，7按顺序依次进栈，则这7个元素的出栈顺序可能为（ AD ）

A. 1, 2, 3, 4, 5, 6, 7

B. 1, 4, 3, 5, 7, 2, 6

C. 1, 3, 2, 4, 7, 5, 6

D. 2, 3, 7, 6, 5, 4, 1



## 试题（初赛）

- ◎ 设栈s和队列Q初始状态为空,元素1, 2, 3, 4, 5, 6依次通过栈s,一个元素出栈后即进入队列Q,若出队的顺序为2, 4, 3, 6, 5, 1, 则栈s的容量至少应该为 ( B ).
- A) 2      B) 3      C) 4      D) 5

# stack

## stack 栈

stack (栈)。是项的有限序列，并满足序列中被删除、检索和修改的项只能是最近插入序列的项。即按照后进先出的原则。

**stack<数据类型> 栈名;** 声明栈，例如：  
**stack<int>s;**

常用函数：

**push(e)** - 将元素e从栈顶压入栈，例如：s.push(5);

**pop()** - 删除栈顶元素，例如：s.pop();

**top()** - 返回栈顶元素的值，例如：cout<<s.top();

**size()** - 栈中元素的个数，例如：cout<<s.size();

**empty()** - 判断栈是否为空，例如：s.empty();

# stack 栈

```
int main ()
{
    stack<int> sk;                //申明一个int类型的stack变量sk
    int sum =0;
    for (int i=1;i<=10;i++) sk.push(i);    //将数字1到10入栈
    cout<<sk.size();                //输出栈中元素个数10
    while (!sk.empty())            //只要栈不为空
    {
        sum += sk.top();            //将栈顶元素的值累加
        sk.pop();                    //栈顶元素出栈
    }
    cout << sum << endl;
}
```

# 栈的实例3：括号匹配问题

我们做小学数学题时，数学式子里往往会加很多括号，有时候这些括号可能出错。比如下面式子：

$32 \times [ 56 + ( 72 - 65 ] \times 25 )$                       括号不匹配

$32 \times [ 56 + ( 72 - 65 \times 25 ) ]$                       括号匹配

现在我们只关心括号是否匹配，所以，我们可以把式子中的括号单独提出来

[ ( ] )

[ ( ) ]

对于给出的由 '{', '}', '[', ']', '(', ')' 构成的长度不超过100000的括号序列，请你快速回答里面的括号是否匹配

**输入格式：**

一行，一个括号序列

**输出格式：**

一行，一个单词 "yes" 或者 "no"

**样例输入1：**

( [ { ( ) [ ] } [ ( ) ] ( ) ] )

**样例输出1：**

yes

**样例输入2：**

( [ { ( [ ] ) } [ ( ) ] ( ) ] )

**样例输出2：**

no

## 栈的实例3：括号匹配问题

解法：从左往右扫描每一个括号，若遇到一个“右”括号，那么它一定与左边离它最近的一个还未配对的“左”括号配对。若配对成功，则将这对括号删除，否则说明配对出错。

```
int main()
{
    stack<char>A;
    string s;
    cin>>s;
    int n=s.length();
    for(int i=0;i<n;i++)
    {
        if(s[i]=='[' || s[i]=='(' || s[i]=='{')A.push(s[i]);
        else
        {
            if(A.size()==0){ cout<<"no"; return 0; }
            if( (s[i]==']' && A.top()=='[') || (s[i]==')' && A.top()=='(') || (s[i]=='}' && A.top()=='{') )A.pop();
            else { cout<<"no"; return 0; }
        }
    }
    if(A.size()==0)cout<<"yes";
    else cout<<"no";
    return 0;
}
```

## 栈的实例4：火车调度 nkoj1914

某城市有一个火车站，如下图 所示，现有  $n$  ( $n \leq 10000$ ) 节火车车厢，顺序编号为  $1, 2, 3, \dots, n$ ，按编号连续依次从 A 方向的铁轨驶入车站，从 B 方向铁轨驶出。一旦车厢进入车站就不能再回到 A 方向的铁轨上；在车站的门口有工人可以将车厢拖出车站，工人一次只能拖一节车厢，并且只能将车厢拖入B方向的铁轨。一旦车厢出了车站就不能再回到车站。车站一开始为空，最多能停放 10000 节车厢。

为了方便装货，调度员需要将车厢重新排列，问能否将车厢编号排列成  $A_1, A_2, \dots, A_n$ 。也就是使车厢从B方向驶出的编号是  $A_1, A_2, \dots, A_n$ 。如果能输出 "yes"，否则输出 "no"。

输入格式：

第一行，一个整数  $n$

第二行， $n$  个用空格间隔的整数，表示出站时车厢编号要排列成的顺序  $A_1, A_2, \dots, A_n$

输出格式：

一行，一个单词 "yes" 或者 "no"

样例输入1：

5

3 2 5 4 1

样例输出1：

yes

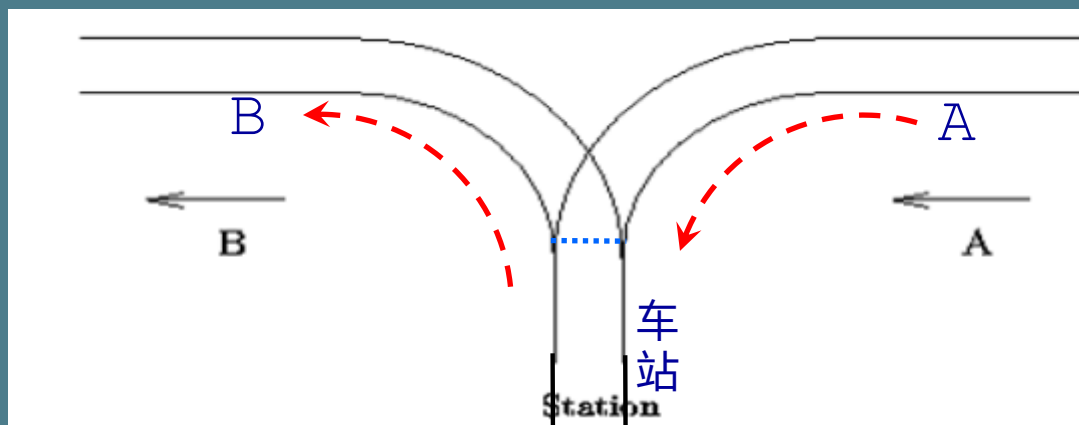
样例输入2：

5

3 1 5 4 2

样例输出2：

no



## 课后练习：网页浏览 NKOI1085

网页浏览器都包含有前进和后退功能，以此来快速打开之前你访问过的网页。你的任务就是实现这个功能。实现此功能的常用方法是通过forward和back两个栈来保存前进和后退时要打开的网页。下列命令是你需要实现的：

**BACK**：把当前显示的网页压入到forward栈中。然后把back栈栈顶的网页弹出作为当前显示的网页。如果back栈为空，那么这条命令就不执行。

**FORWARD**：把当前显示的网页压入到back栈中。然后把forward栈栈顶的网页弹出作为当前显示的网页。如果forward栈为空，那么这条命令就不执行。

**VISIT**：把当前显示的网页压入到back栈中。然后浏览器显示用户新输入的网址对应的网页。清空forward栈

**QUIT**：关闭浏览器

假设只要浏览器一打开就会自动打开网页http://www.acm.org/

**输入格式：**

包含若干条命令，这些命令由BACK, FORWARD, VISIT和QUIT构成。每条命令占一行，长度不超过70。命令的总条数不超过100。一旦出现QUIT表示结束输入命令。

**输出格式：**

对于每个命令，如果它不是QUIT，那么输出命令执行后浏览器显示的网页。如果该命令无法执行，则输出 "Ignored"，除QUIT命令外，每一个输入命令对应一行输出结果。

### 样例输入

```
VISIT http://acm.ashland.edu/
VISIT http://acm.baylor.edu/acmicpc/
BACK
BACK
BACK
FORWARD
VISIT http://www.ibm.com/
BACK
BACK
FORWARD
FORWARD
FORWARD
QUIT
```

### 样例输出

```
http://acm.ashland.edu/
http://acm.baylor.edu/acmicpc/
http://acm.ashland.edu/
http://www.acm.org/
Ignored
http://acm.ashland.edu/
http://www.ibm.com/
http://acm.ashland.edu/
http://www.acm.org/
http://acm.ashland.edu/
http://www.ibm.com/
Ignored
```