第4周小结

1 栈

❶ 先进后出表。1，2，3，…，n通过一个栈的出栈序列个数？

$$\frac{1}{n+1}C_{2n}^{n}=\frac{1}{n+1}\times\frac{(2n)!}{n!\times n!}$$

第n个Catalan数

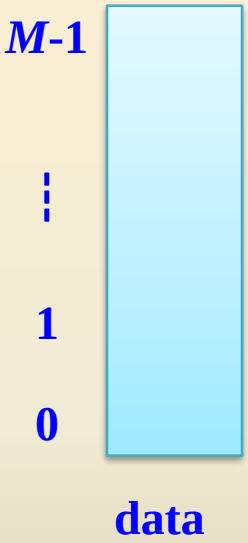
当n=3，结果为5。

❷ 一个大小为n的顺序栈，最多只能进行n次进栈操作吗？

错误： 可以进行任意多次进栈操作。

但最多只能进行连续n次进栈操作。

❸ 顺序栈只能将栈底设置在data[0]端吗？

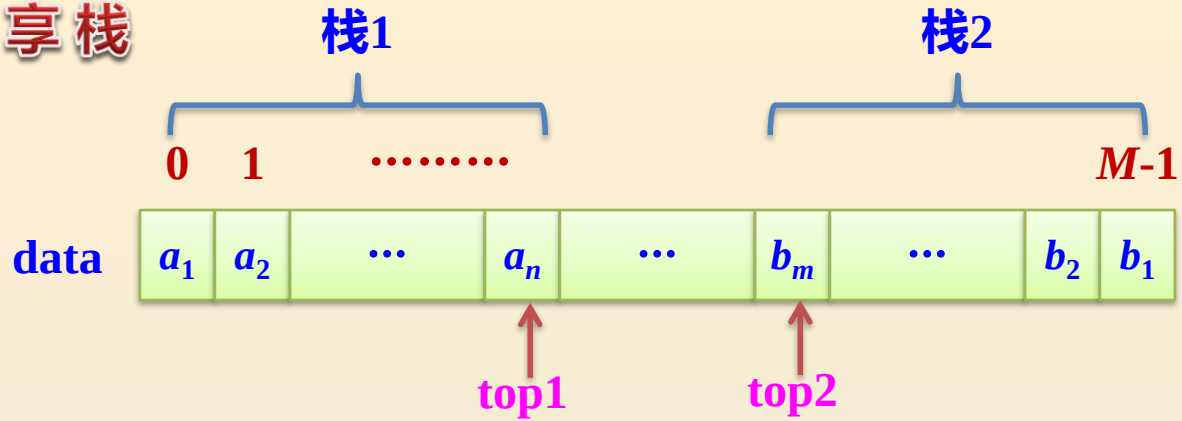


可以将栈底设置在任意一端，但不能设置在中间。

将栈底设置在data[*M*-1]端的设计：

- 初始化：　top=*M*
- 栈空：　top==*M*
- 栈满：　top==0（最小下标）
- 元素*e*进栈：top--; data[top]=*e*
- 出栈：　*e*=data[top]; top--

❹ 共享栈



- 初始化： top1=-1; top2=*M*;
- 栈1空： top1==-1　栈2空： top2==*M*
- 栈满： top1+1=top2
- 元素进栈1： top1++; data[top1]=*e*;
- 元素进栈2： top2--; data[top2]=*e*;
- 栈1出栈： *e*=data[top1]; top1--;
- 栈2出栈： *e*=data[top2]; top2++;

2 队列

❶ 先进先出表。1，2，3，…，*n*通过一个队列的出队序列个数？

只有一个：即1，2，3，…，*n*

❷ 环形队列解决了假溢出问题，任何情况下都使用环形队列吗？

- 采用环形队列时，进队的元素可能被覆盖。
- 如果需要用队列中全部进队的元素进一步求解问题，应该采用非环形队列。如用队列求解迷宫路径！

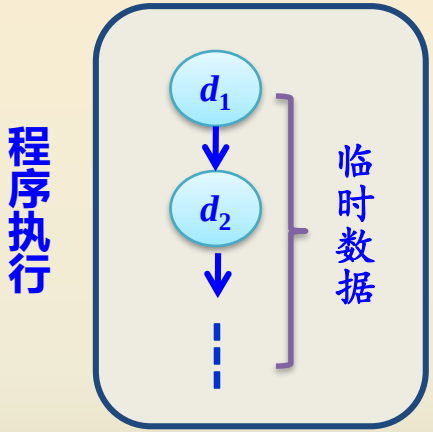
❸ 如果需要多个队列，可以像共享栈一样设置共享队列吗？如果需要使用10个队列，如何设计？

不能。因为栈是向一端生长的，而队列不是。为了节省空间，应该采用链队。

如果需要使用10个队列，可以设置10个链队：

- 队头指针：front[10]
- 队尾指针：rear[10]

3 栈和队列的应用



保存d_1、d_2、…

- 先产生的数据后处理—栈（先进后出表）
- 先产生的数据先处理—队列（先进先出表）

简单表达式求值

这里限定的简单表达式求值问题是：用户输入一个包含“+”、“-”、“*”、“/”、正整数和圆括号的合法算术表达式，计算该表达式的运算结果。

例如，*exp*=“1+2*(4+12)” ⇦ 中缀表达式

中缀表达式的运算规则：“先乘除，后加减，从左到右计算，先括号内，后括号外”。

因此，中缀表达式不仅要依赖运算符优先级，而且还要处理括号。

算术表达式的另一种形式是后缀表达式或逆波兰表达式，就是在算术表达式中，运算符在操作数的后面。

如1+2*3的后缀表达式为1 2 3 * +。

后缀表达式：

- 已考虑了运算符的优先级。
- 没有括号。
- 只有操作数和运算符，而且越放在前面的运算符来越优先执行。

中缀表达式的求值过程：

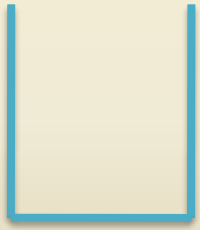
- 将中缀算术表达式转换成后缀表达式。
- 对该后缀表达式求值。

（1）将算术表达式转换成后缀表达式

exp ⇒ *postexp*

扫描*exp*的所有字符：

- **数字字符直接放在*postexp*中**
- **运算符通过一个栈来处理优先级**

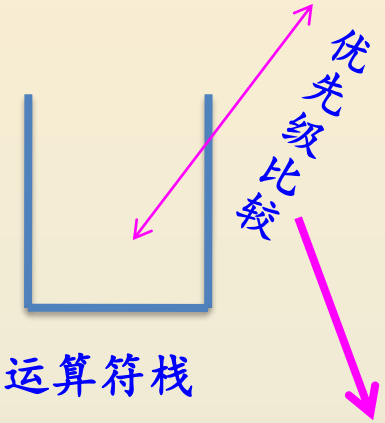


运算符栈

exp ⇒ *postexp*

情况1（没有括号）

exp= “ 1 + 2 + 3 ”



postexp：

“1+2+3” ⇒ “1 2 + 3 +”

- 先进栈的先退栈即先执行：

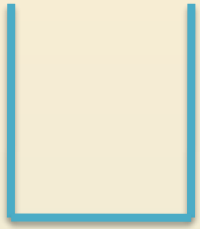
只有大于栈顶优先级才能直接进栈

- *exp*扫描完毕，所有运算符退栈

exp ⇒ *postexp*

情况2（带有括号）

exp= “ 2 * (1 + 3) − 4 ”



运算符栈

postexp：

postexp=“2 1 3 + * 4 −”

- 开始时，任何运算符都进栈
- (：一个子表达式开始，进栈
- 栈顶为(：任何运算符进栈
-)：退栈到 (
- 只有大于栈顶的优先级，才进栈；否则退栈

while (从exp读取字符ch，ch!='\0')

{ ch为数字：将后续的所有数字均依次存放到postexp中，

并以字符'#'标志数值串结束;

ch为左括号'('：将此括号进栈到Optr中;

ch为右括号')'：将Optr中出栈时遇到的第一个左括号'('以前的运算符依次出

栈并存放到postexp中，然后将左括号'('出栈;

ch为其他运算符：

if （栈空或者栈顶运算符为'('） 直接将ch进栈;

else if (ch的优先级高于栈顶运算符的优先级)

直接将ch进栈;

else

依次出栈并存入到postexp中，直到栈顶运算符优先级小于ch的优先级，

然后将ch进栈;

}

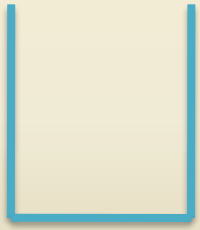
若exp扫描完毕，则将Optr中所有运算符依次出栈并存放到postexp中。

（2）后缀表达式求值

postexp ⇒ 值

扫描*postexp*的所有字符：

- **数字字符：转换为数值并进栈**
- **运算符：退栈两个操作数，计算，将结果进栈**



操作数栈

while (从postexp读取字符ch，ch!='\0')

{　　ch为'+'：从Opnd栈中出栈两个数值*a*和*b*，计算*c*=*b*+*a*;将*c*进栈;

　　ch为'-'： 从Opnd栈中出栈两个数值*a*和*b*，计算*c*=*b*-*a*;将*c*进栈;

　　ch为'*'：从Opnd栈中出栈两个数值*a*和*b*，计算*c*=*b***a*;将*c*进栈;

　　ch为'/'： 从Opnd栈中出栈两个数值*a*和*b*，若*a*不为零，计算*c*=*b*/*a*;将*c*进栈;

　　ch为数字字符：将连续的数字串转换成数值*d*，将*d*进栈;

}

返回Opnd栈的栈顶操作数即后缀表达式的值;