# 4.2 递归及递归算法

# 课程安排



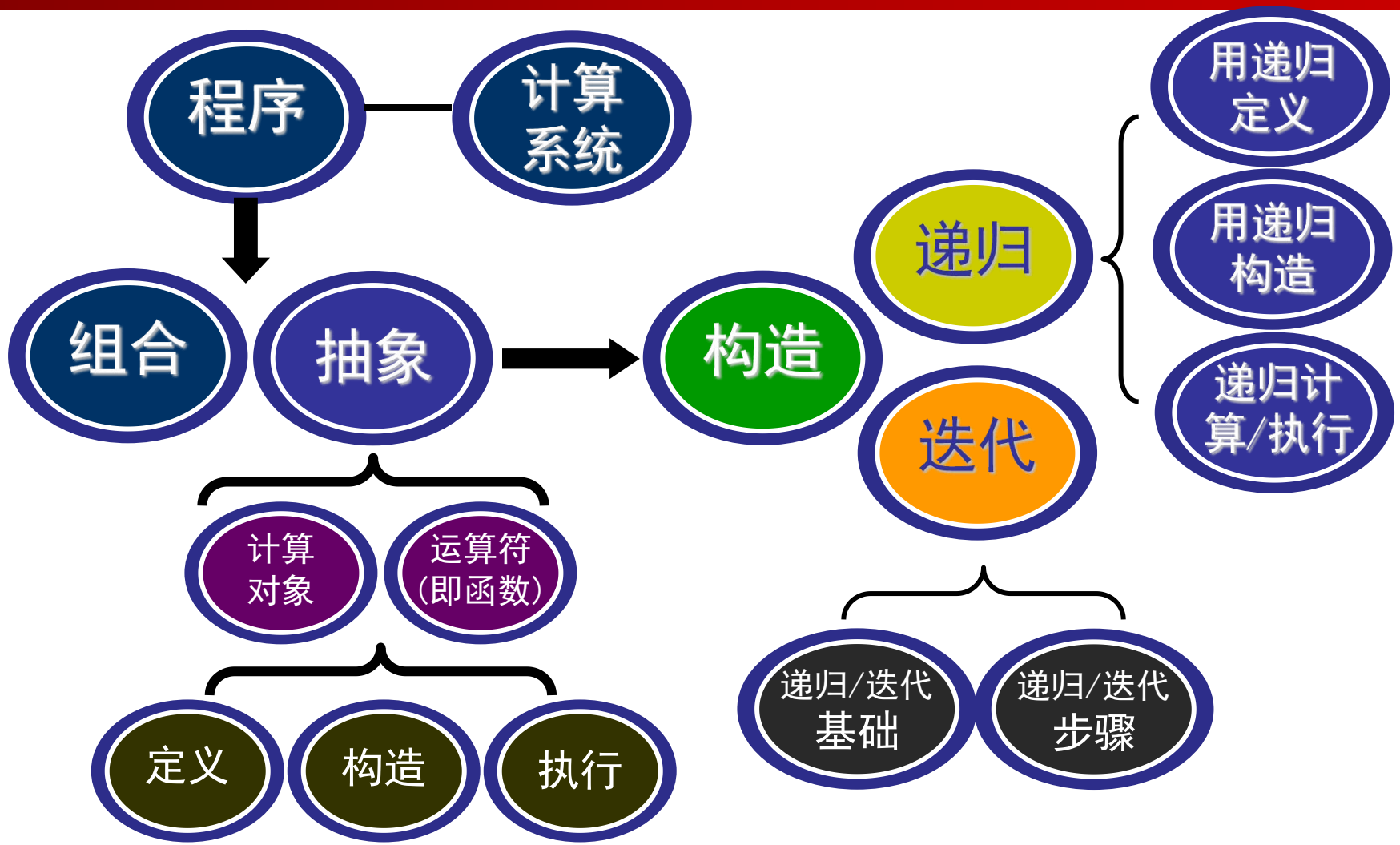
## 主要内容

计算系统与程序
运算式的组合-抽象与构造
程序构造的基本手段：递归与迭代
- 递归的概念
- 原始递归函数-复合与递归
- 两种不同的递归函数-递归与迭代
- 运用递归与迭代
- 递归与迭代程序的执行

# 组合、抽象与构造的程序思维概述



## 程序构造思维概述



# 计算系统与程序



## 怎样构造一台计算机器?

**已知：（1）“加减乘除运算都可转换为加减法运算来实现”**

**（2）“加减法运算又可以转换为逻辑运算来实现”**

首先，设计并实现系统可以执行的基本动作(可实现的)，例如：

**“与”**动作、 **“或”**动作、 **“非”**动作、 **“异或”**动作


那么，复杂的动作呢？

# 计算系统与程序



## 构造计算机器的基本思维：程序

程序：由基本动作指令构造的，若干指令的一个组合或一个执行序列，用以实现复杂动作

复杂动作

((A AND B) AND C) OR (NOT C))

拆解开

X= A AND B
X= X AND C
Y= NOT C
X= X OR Y

系统

“与”动作 — AND

“或”动作 — OR

“非”动作 — NOT

指令：控制基本动作执行的命令

# 计算系统与程序



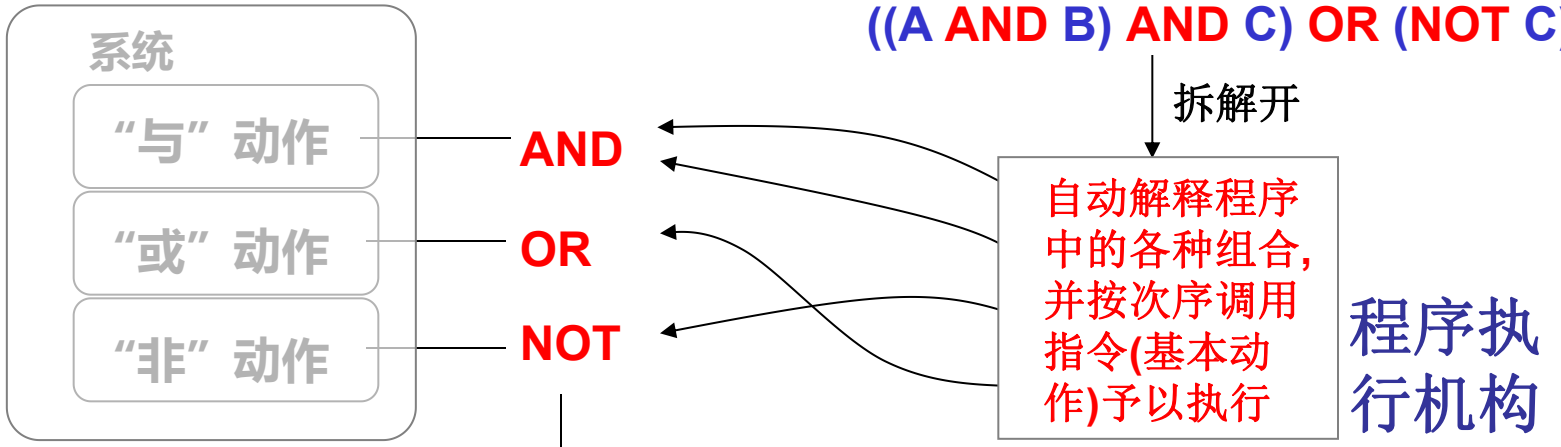
## 自动计算机器：程序 与 程序执行机构

程序：由基本动作指令构造的，若干指令的一个组合或一个执行序列，用以实现复杂动作

复杂动作

((A AND B) AND C) OR (NOT C))

拆解开




指令：控制基本动作执行的命令

# 计算系统与程序



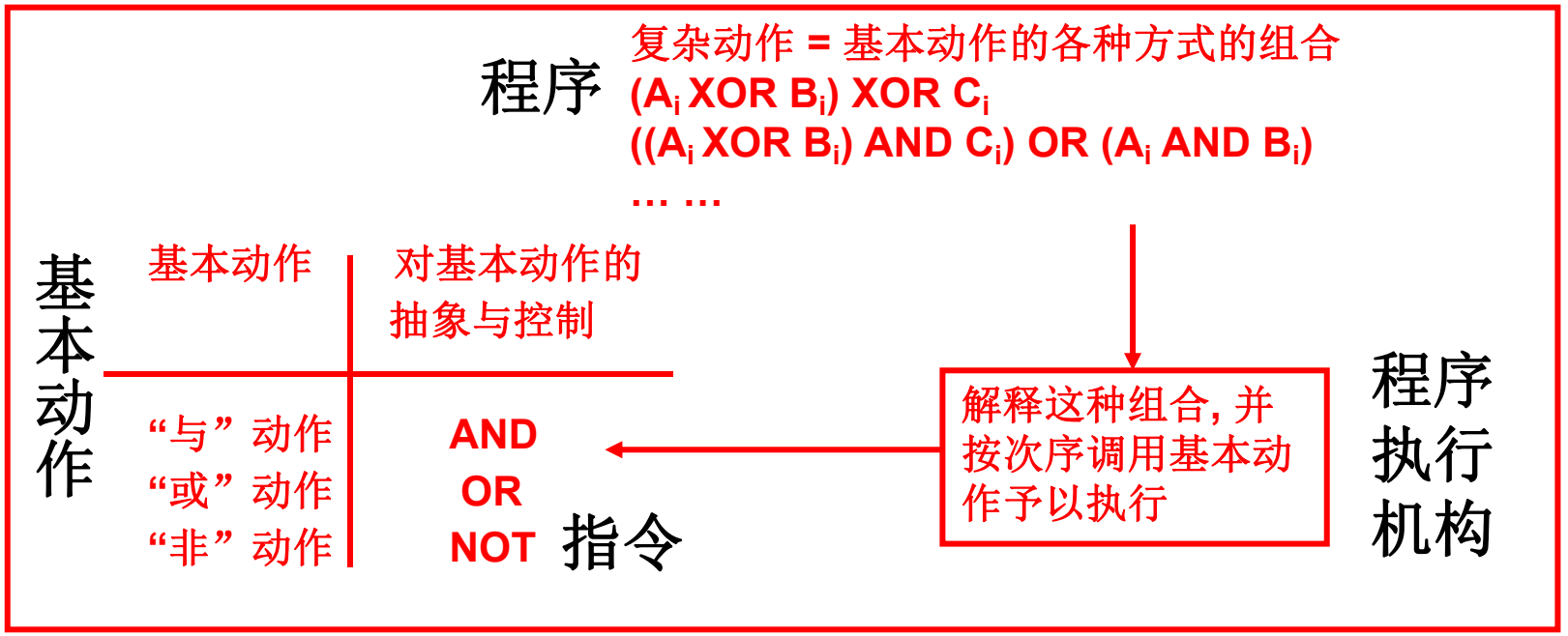
## 自动计算机器：程序 与 程序执行机构

**计算系统** = 基本动作 + 指令 + 程序执行机构

**指令** = 对可执行基本动作的抽象，即控制基本动作执行的命令

**程序** = 基本动作指令的一个组合或执行序列，用以实现复杂的动作

**程序执行机构** = 负责解释程序即解释指令之间组合并按次序调用指令即调用基本动作执行的 机构



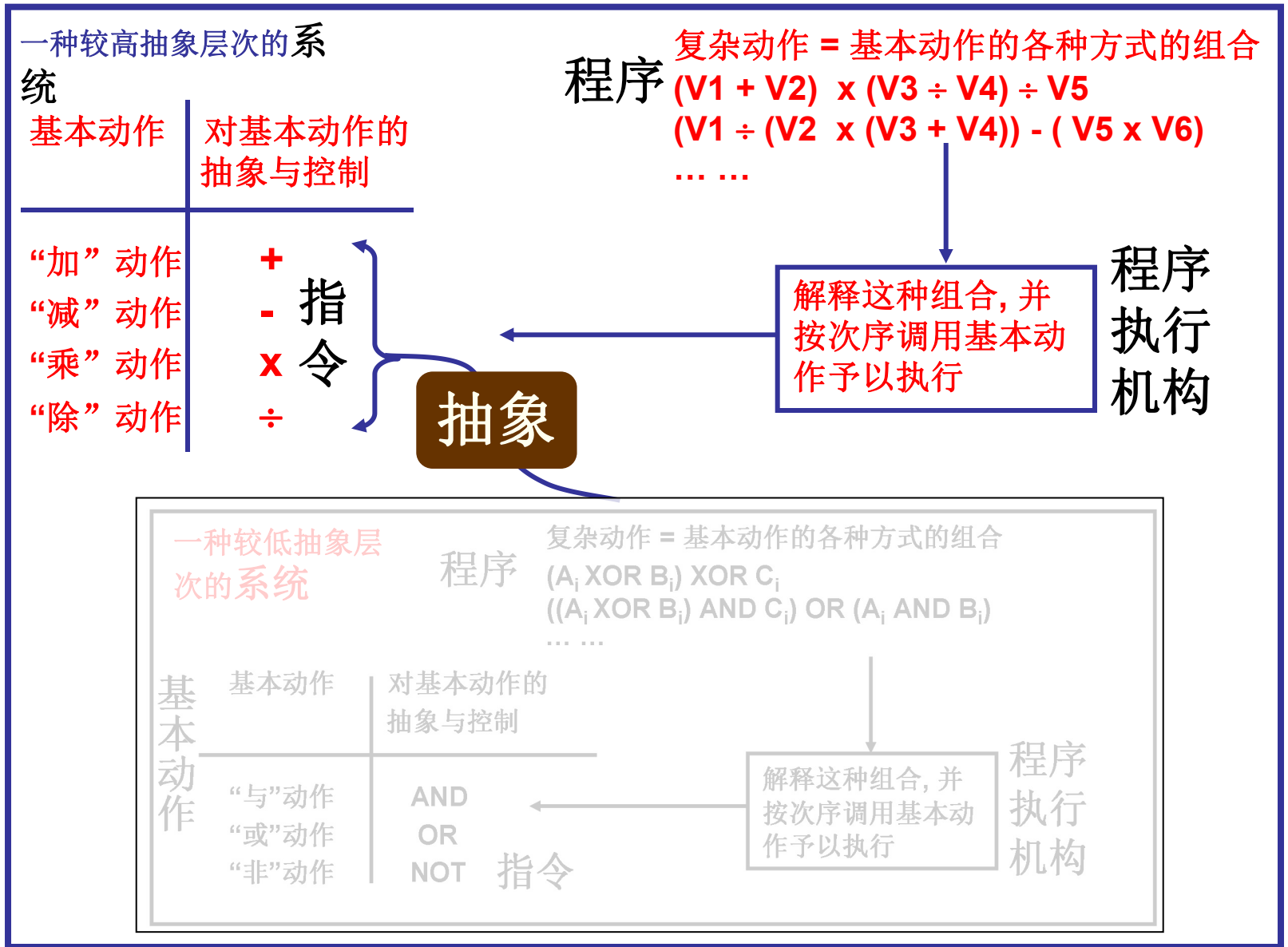
# 计算系统与程序



## 分层式构造

高抽象层次vs. 低抽象层次

**抽象**：将经常使用的、可由低层次系统实现的一些复杂动作，进行**命名**，以作为高层次系统的指令被使用



单选题 1分

设置

9

一般而言，设计和实现一个计算系统，需要设计和实现_____。

A 基本动作和程序；

B 基本动作和控制基本动作的指令；

C 基本动作、控制基本动作的指令和一个程序执行机构；

D 基本动作、控制基本动作的指令和程序

提交

# 运算式的组合-抽象与构造---程序构造示例



## 运算组合式——种程序的表达方法

由数值，到基本运算组合式

*100*

*205*

*实际的数值*

*(100 + 205)*

*中缀表示法，用运算符(即前述的指令)将两个数值组合起来，运算符在中间*

*(+ 100 205)*

*前缀表示法，用运算符(即前述的指令)将两个数值组合起来，运算符在前面*

*将运算符表示的操作应用于后面的一组数值上，求出结果*

*(+ 100 205 300 400 51 304)*

*一个运算符可以表示连加，连减等情况，*

# 运算式的组合-抽象与构造---程序构造示例



## 运算组合式——种程序的表达方法

由数值，到基本运算组合式

*(+ 100 205)*

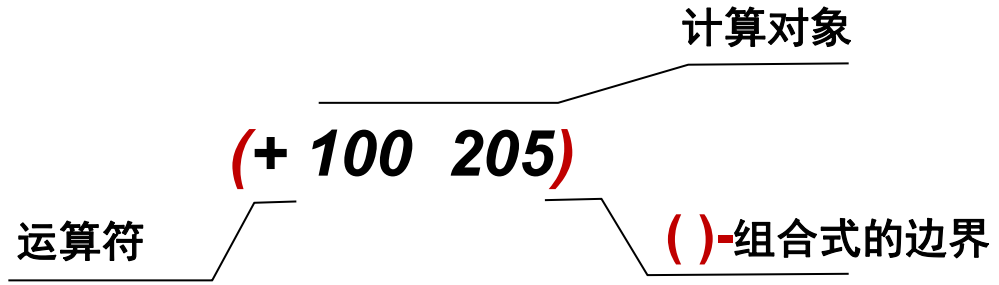
*(- 200 50)*

*(* 200 5)*

*(* 20 5 4 2)*

*(- 20 5 4 2)*

*(+ 20 5 4 2)*




- 一个组合式内只能有一个运算符，可有多个计算对象
- 基本运算符只有+、-、*、/，可以被识别和计算。

*一起练习,书写程序,…… ……*

# 运算式的组合-抽象与构造---程序构造示例



## 运算组合式的“组合-嵌套”（构造）及其计算过程（执行）

*(+ 100 205)*

*(+ (+ 60 40) (- 305 100))*

*(* (* 3 (+ (* 2 4) (+ 3 5))) (+ (- 10 7) 6))*

(运算符1 计算对象1 计算对象2)

(运算符2 计算对象3 计算对象4)

(运算符1 (运算符2 计算对象3 计算对象4) 计算对象2)

*计算过程*

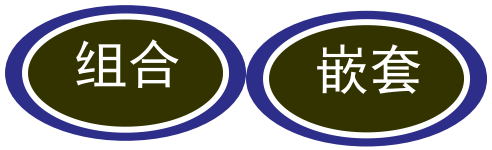
*(* (* 3 (+ (* 2 4) (+ 3 5))) (+ (- 10 7) 6))*

*(* (* 3 (+ 8 8)) (+ 3 6))*

*(* (* 3 16) 9 )*

*(* 48 9 )*

*432*



# 运算式的组合-抽象与构造---程序构造示例



## 用名字替代计算对象（构造）

命名计算对象和构造中使用名字及计算中以计算对象替换名字

*(define height 2)* ——— 名字的定义：定义名字height与2关联，以后可以用height来表示2
一种类型的名字：数值型的名字


*(+ (+ height 40) (- 305 height))*

*(+ (* 50 height) (- 100 height))*

——— 名字的使用

注意：不同类型的对象可以有不同的定义方法。这里统一用define来表示，在具体的程序设计语言中是用不同的方法来定义的

# 运算式的组合-抽象与构造---程序构造示例



## 用名字替代计算对象（构造）

```
(define pi 3.14159)
(define radius 10)
(*  pi  (* radius radius))
```

```
(*  pi  (*  radius radius))
(*  pi  (* 10  10))
(*  pi  100)
(* 3.14159 100)
314.159
```
计算

```
(define  circumference  (* 2 pi radius))
(*  circmference  20)
```

```
(*  circmference  20)
(* (* 2  pi  radius) 20)
(* (* 2  3.14159  10) 20)
(* 62.8318  20)
1256.636
```
计算

# 运算式的组合-抽象与构造---程序构造示例



## 定义新运算符与使用新运算符（构造）

**命名新运算符和构造中使用新运算符及执行中以过程替换新运算符**

*(define (square x) (* x x))*

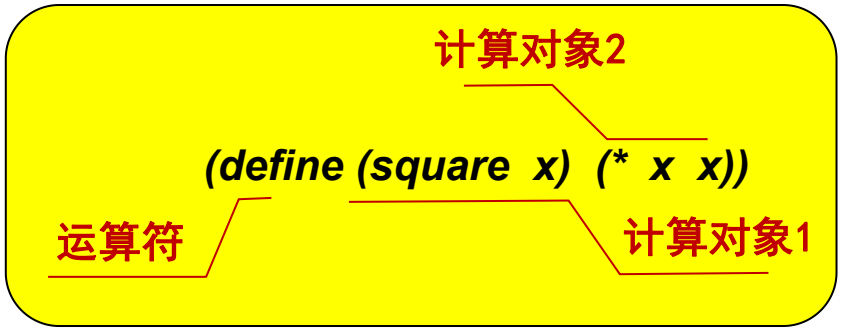
$x^2$

名字的定义：定义名字**square**为一个新的运算，即**过程**或称**函数**。另一种类型的名字：运算符型的名字

新运算符，*即过程名或函数名*

**形式参数，**使用时将被实际参数所替代

过程体，用于表示新运算符的具体计算规则，其为关于形式参数**x**的一种计算组合。




*(square 3)*
*(square 6)*

名字的使用

*注意：不同类型的对象可以有不同的定义方法。这里统一用define来表示，在具体的程序设计语言中是用不同的方法来定义的*

# 运算式的组合-抽象与构造---程序构造示例



## 新运算符定义与构造示例

```
(square  10)
(square  (+ 2 8))
(square  (square 3))
(square  (square (+ 2 5)))
```

```
(define (SumOfSquare  x  y)  (+  (square x) (square y)) )
```

$x^2+y^2$

```
(SumOfSquare  3  4)
(+ (SumOfSquare  3  4)  height)
```

# 运算式的组合-抽象与构造---程序构造示例



## 新运算符定义与构造示例

*(define (NewProc a) (SumOfSquare (+ a 1) (* a 2)))*

$(a+1)^2+(a*2)^2$

*(NewProc 3)*

*(NewProc (+ 3 1))*

# 运算式的组合-抽象与构造---程序构造示例



## 运算组合式的执行（计算方法1）

### 求值、代入、计算

```
(NewProc (+  3  1))
(NewProc 4)
(SumOfSquare (+  4  1) (*  4  2))
(SumOfSquare 5  8)
(+  (Square  5)  (Square  8))
(+  (*  5  5)  (*  8  8))
(+  25  64)
89
```

先求值，再代入

# 运算式的组合-抽象与构造---程序构造示例



## 运算组合式的执行（计算方法2）

### 代入、求值、计算

代入到只有基本运算符时再开始求值

```
(NewProc (+  3  1))
(SumOfSquare(+  (+ 3  1)  1) (*  (+ 3 1)  2))
(+  (Square  (+ (+ 3  1)  1))  (Square  (* (+  3  1) 2)))
(+  (*  (+ (+ 3  1)  1)  (+ (+ 3  1)  1)) (*  (* (+  3  1) 2) (* (+  3  1) 2)))
(+  (*  (+ 4  1)  (+ 4 1)) (*  (* 4 2) (* 4 2)))
(+  (*  5  5) (*  8  8))
(+  25  64)
89
```

先代入，后求值

代入阶段

求值阶段

# 运算式的组合-抽象与构造---程序构造示例



## 构造与执行练习

对于计算式，其正确的运算组合式(前缀表示法)为_____。

(A) (/ (+ 10 / 20 + 8 4) (+ * 3 6 * 8 2 ));

(B) ((10 + (20 / (8 + 4))) / ((3 * 6) + (8 * 2)));

(C) (/ (+ 10 (/ 20 (+ 8 4))) (+ (* 3 6) (* 8 2)));

(D) (/ (/ 20 (+ 10 (+ 8 4))) (* (+ 3 6) (+ 8 2)))。

$$\frac{10+\frac{20}{8+4}}{3*6+8*2}$$

( / 操作数1 操作数2)

( / (+ 10 操作数a2) (+ 操作数a3 操作数a4 ) )

( / (+ 10 (/ 20 (+ 8 4) )) (+ 操作数a3 操作数a4 ) )

( / (+ 10 (/ 20 (+ 8 4) )) (+ (* 3 6) (* 8 2) ) )

单选题 1分

设置

21

已知一个新运算被定义为(define (newCalc x y) (* (+ x 1) (* y 2)))，问正确使用了newCalc并得到正确结果的为_____。

A ((newCalc) (4 5))，其结果为50；

B (newCalc 4)，其结果为40；

C (newCalc 4 5)，其结果为50；

D (newCalc 2 3)，其结果为21

提交

# 运算式的组合-抽象与构造---程序构造示例



## 构造与执行练习

◆问题1：用前缀表示法书写下述表达式

$$\frac{10 + 4 + (8- (12 - (6 + 4 \div 5)))}{3*(6-2)(12-7)}$$

◆问题2：请定义一个过程，求某一数值的立方 $a^3$

◆问题3：进一步以问题2定义的过程，再定义一个过程，求某两个数值的立方和 $a^3+b^3$ 。 进一步求 $5^3+8^3$ ，并模拟给出计算过程。

# 运算式的组合-抽象与构造---程序构造示例



## 带有条件的运算组合式如何表达

◆问题4：请定义一个过程，计算下列函数

$$f(x)=\begin{cases} x^2-x & if\ x>0 \\ 0 & if\ x=0 \\ -x^2+x & if\ x<0 \end{cases}$$

```
(cond  ( <p1>  <e1>)
       ( <p2>  <e2>)
       ...
       ( <pn>  <en>) )
```

```
(define  (f  x) (cond  ((>  x  0)  (-  (Square x)  x))
                       ((=  x  0)  0)
                       ((<  x  0)  (-  x  (Square x) ))  ))
```

# 递归的概念



## 为什么需要递归?

怎样表达n!的计算过程

```
(* … (* (* (* 1 1) 2) 3)…n)
```

怎样在表达中既去掉省略号，而又能表达近乎无限的内容?

# 递归的概念



## 为什么需要递归?

### 计算规则的构造与正确性判断

*(运算符1 操作数1 操作数2)*

*(运算符2 操作数a 操作数b)*

*(运算符1 (运算符2 操作数a 操作数b) 操作数2)*

*( / (+ 205 (/ (+ 15 3) (- 90 (× 8 8)) )) (- 200 (× 10 5) ) )*

*( / (+ 205 (/ (+ 15 3 - 90 (× 8 8)) )) (- 200 (× 10 5) ) )*

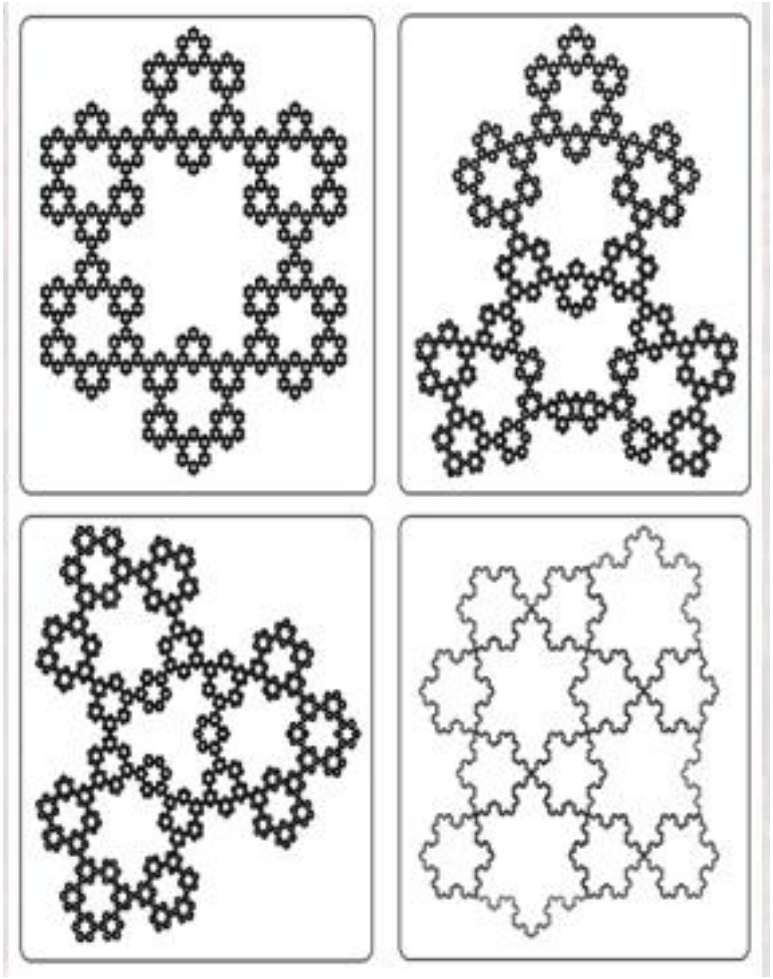

**怎样表达运算组合式的层层构造--构造规则?**

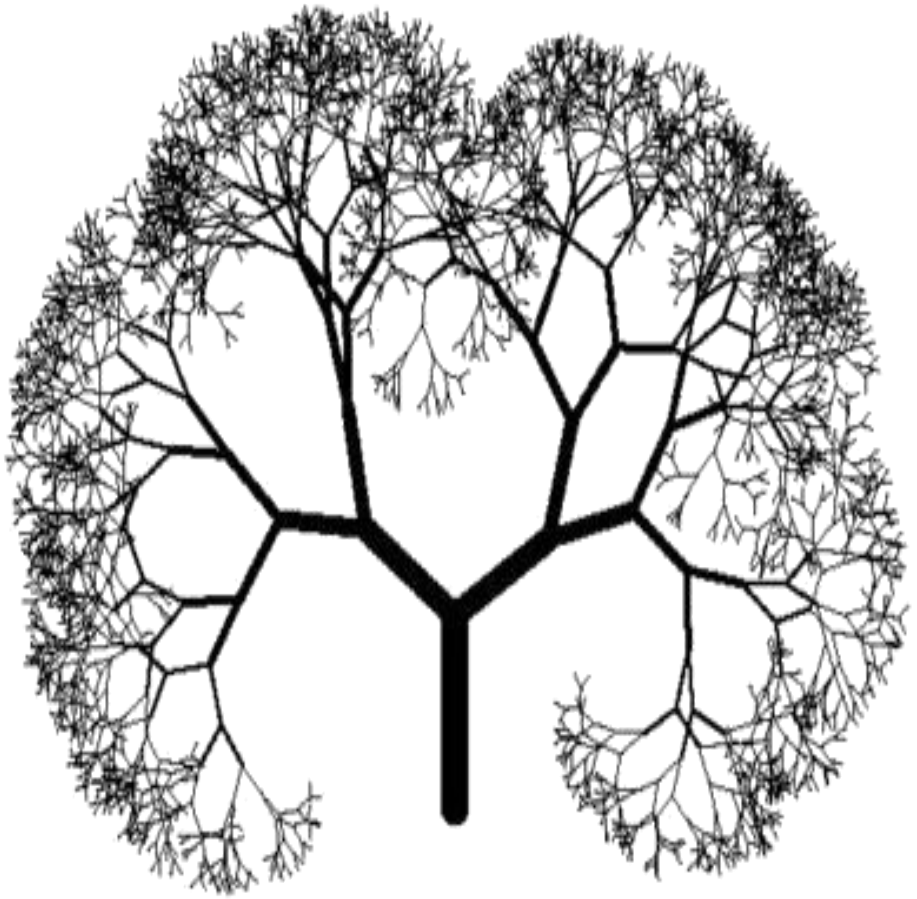
**怎样判断一个运算组合式是否符合构造规则?**

# 递归的概念



## 递归的典型视觉形式---自相似性事物的无限重复性构造



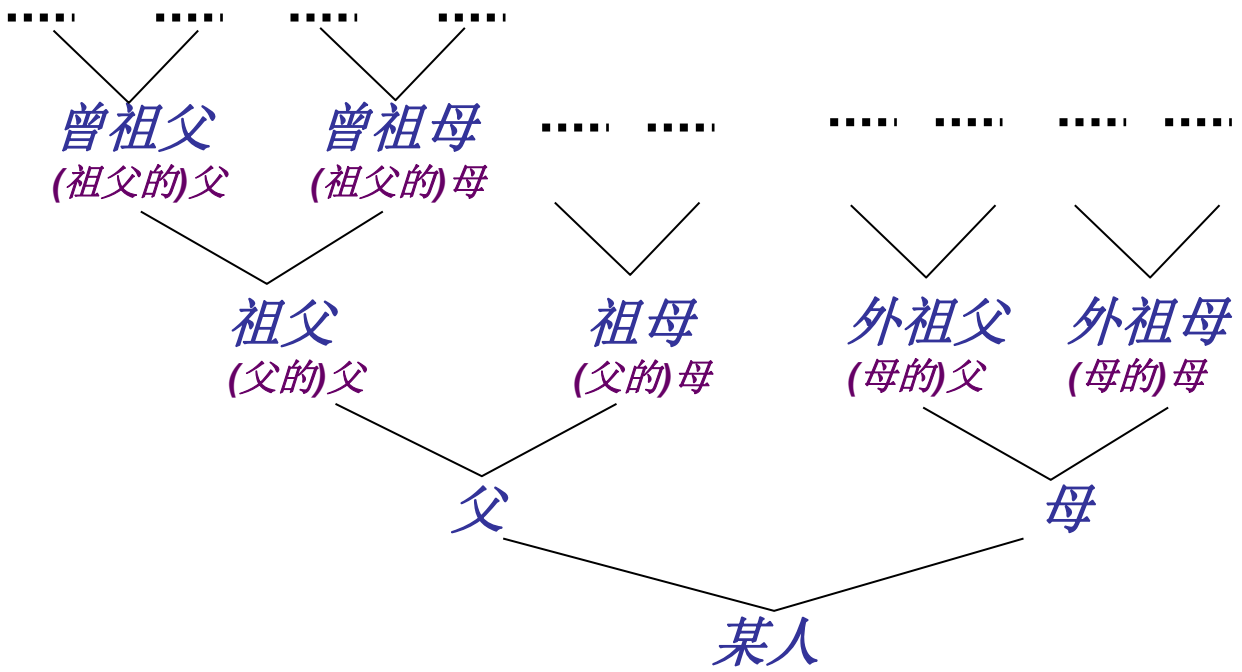


# 递归的概念



## 怎样递归?

怎样定义你的祖先?



# 递归的概念



## 怎样递归?

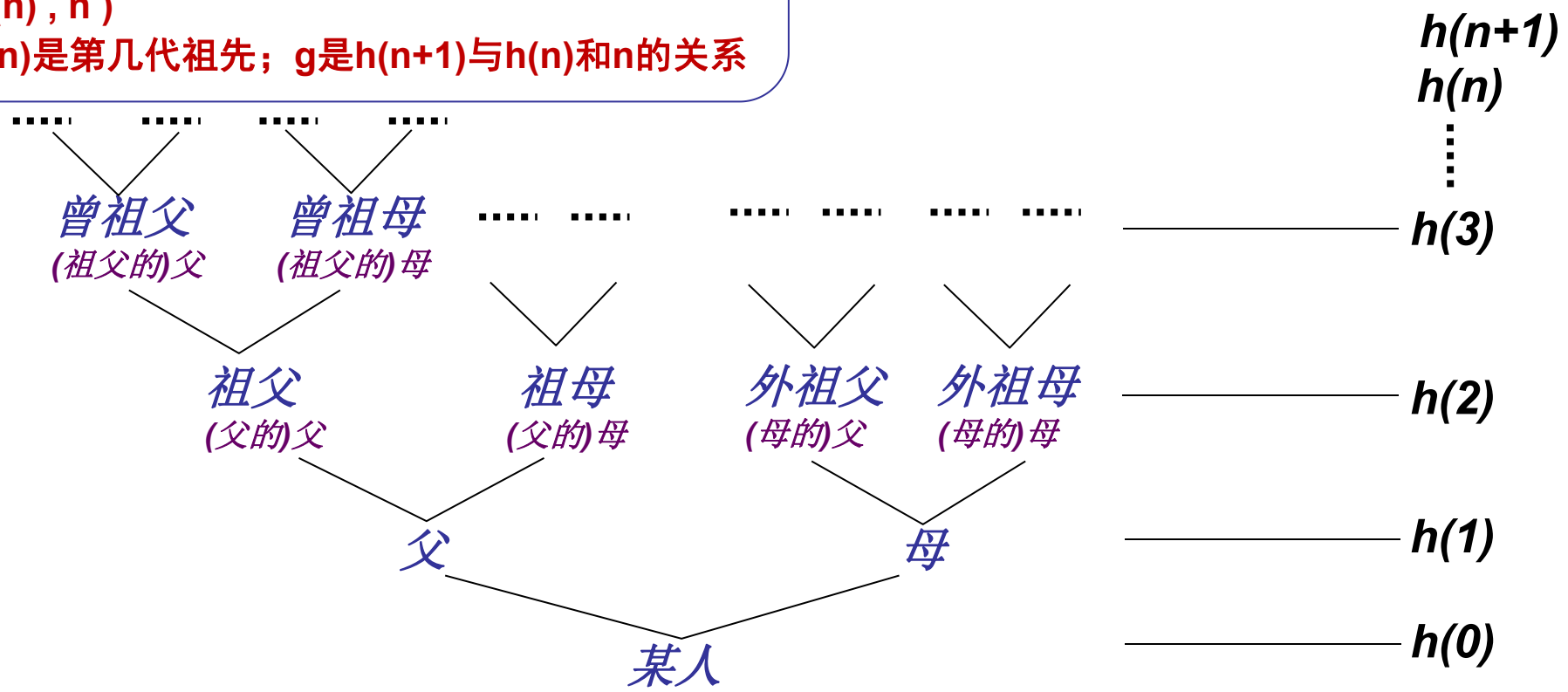
(1)某人的双亲(父母)是他的祖先（递归基础）

--- h(1)直接给出

(2)某人祖先的双亲(父母)同样是某人的祖先（递归步骤）

--- h(n+1) = g( h(n) , n )

--- n: 第几代；h(n)是第几代祖先；g是h(n+1)与h(n)和n的关系



# 递归的概念



## 数学的递推式

一个数列的第n项$a_n$与该数列的其他一项或多项之间存在某种对应关系，被表达为一种公式，称为递推式

*等差数列递推公式*

$a_0=5$

$a_n=a_{n-1}+3$　　*当n>=1时*

- 第1项(或前K项)的值是已知的—递推基础；
- 由第n项或前n项计算第n+1项—递推规则/递推步骤；
- 由前向后，可依次计算每一项

等差数列的产生

$a_0=5$

$a_1=a_0+3 = 8$

$a_2=a_1+3 = 11$

$a_3=a_2+3 = 14$

$a_4=a_3+3 = 17$

… …

# 递归的概念



## 运算组合式的递归定义示例

( / (+ 10 4 (- 8 (- 12 (+ 6 (/ 4 5))))) (- 15 8) )

是运算组合式吗?

(1)单一数值是运算组合式

--- h(0)直接给出，递归基础

(2)如果X, Y是运算组合式，则

(+ X Y), (* X Y), (- X Y),

(/ X Y)也是运算组合式

--- h(n+1) = g( h(n) , n )，即递归步骤

说明：这里不允许连加、连减情况发生

--- n: 第几层；h(n)是第几层表达式；

g是(运算符 操作数1 操作数2)，指出h(n+1)与h(n)和n的关系

h(n+1) —— ( / 操作数1 操作数2)

h(n) —— ( / (+ 10 4 操作数a2) 操作数2 )

h(n-1) —— ( / (+ 10 4 (- 8 操作数a3)) 操作数2 )

⋮

( / (+ 10 4 (- 8 (- 12 操作数a4))) 操作数2 )

h(1) —— ( / (+ 10 4 (- 8 (- 12 (+ 6 操作数a5)))) 操作数2 )

h(0) —— ( / (+ 10 4 (- 8 (- 12 (+ 6 (/ 4 5))))) 操作数2 )

---

h(0) —— (/ 4 5)

h(1) —— (+ 6 (/ 4 5))

h(2) —— (- 12 (+ 6 (/ 4 5)))

⋮

(- 8 (- 12 (+ 6 (/ 4 5))))

h(n) —— (+ 10 4 (- 8 (- 12 (+ 6 (/ 4 5)))))

h(n+1) —— ( / (+ 10 4 (- 8 (- 12 (+ 6 (/ 4 5))))) 操作数2 )

# 递归的概念



## 什么是递归?

递归是表达相似性对象及动作的无限性重复性构造的方法。

■递归基础：定义、构造和计算的起点，直接给出。即h(0)。

■递归步骤：由前n项或第n项定义第n+1项；由低阶h(k)且k<n，来构造高阶h(n+1)；

即: h(n+1) = g ( h(n)，n )，或者 h(n+1) = g( h(k), n, k )形式，g是需要明确给出的，以说明h(n+1)怎样由h(k), k和n构造出来。

- 递归是一种关于抽象的表达方法---用递归定义无限的相似事物
- 递归是一种算法或程序的构造技术---自身调用自身，高阶调用低阶，构造无限的计算步骤
- 递归是一种典型的计算/执行过程---由后向前代入，直至代入到递归基础，再由递归基础向后计算直至计算出最终结果，即由前向后计算

用递归定义

用递归构造

递归计算/执行

# 递归与迭代



## 数学上如何构造递归函数

原始递归函数是接受自然数x或自然数的元组$(x_1,\cdots x_n)$作为参数，并产生自然数的一个映射，记为f(x)或$f(x_1,\cdots x_n)$。接受n个参数的函数称作n元函数。

最基本的原始递归函数，也被称为本原函数有三个：

(1)初始函数：0元函数即常数无需计算；或者常数函数：对于每个自然数n和所有的k, 有$f(x_1,x_2,\cdots,x_K)=n$。

(2)后继函数：1 元后继函数 S，它接受一个参数并返回给出参数的后继数。例如S(1)=2, …, S(x) = x+1, 其中x为任意自然数。

(3)投影函数：对于所有 n≥1 和每个 1≤i≤n 的 i，n 元投影函数 $P_i^n$，它接受 n 个参数并返回它们中的第 i 个参数，即

$$P_i^n(x_1, x_2, \cdots, x_n) = x_i$$

# 递归与迭代



## 数学上如何构造递归函数

(1)复合：给定原始递归函数 $f(x_1,\ldots,x_k)$，和 k 个原始递归函数 $g_1,\ldots,g_k$，则f 和 $g_1,\ldots,g_k$的复合是 函数h, 即

$$h(x_1,\ldots,x_m) = f(g_1(x_1,\ldots,x_m),\ldots,g_k(x_1,\ldots,x_m))$$

简单而言，复合是将一系列函数作为参数代入到另一个函数中，又被称为代入。复合是构造新函数的一种方法。复合是表达组合的一种方法。

# 递归与迭代



## 数学上如何构造递归函数

(2)原始递归：给定原始递归函数 f 和 g，则新函数h可由 f 和 g递归的定义，其中

$h(0,x_1,\ldots,x_k) = f(x_1,\ldots,x_k)$

$h(S(n), x_1,\ldots,x_k) = g(h(n,x_1,\ldots,x_k), n, x_1,\ldots,x_k)$

简单而言，定义新函数h，就是要定义h(0), h(1),…,h(n),…。h(0)直接给出。h(n+1)则由将h(n)和n代入g中来构造。

原始递归和复合是递归地构造新函数的方法，尤其是无限的相似性函数的构造方法。

(* (* …(* (* (* 1 1) 2) 3) … n) S(n))

$g(x_1, x_2) = (* \ x_1 \ S(x_2))$

$h(0) = 1$

$h(S(n)) = g(h(n), n)$

$h(0) = 1$

$h(1) = g(h(0), 0) = (* \ 1 \ 1)$

$h(2) = g(h(1), 1) = (* \ (* \ 1 \ 1) \ 2)$

$h(3) = g(h(2), 2) = ((* \ (* \ 1 \ 1) \ 2) \ 3)$

…

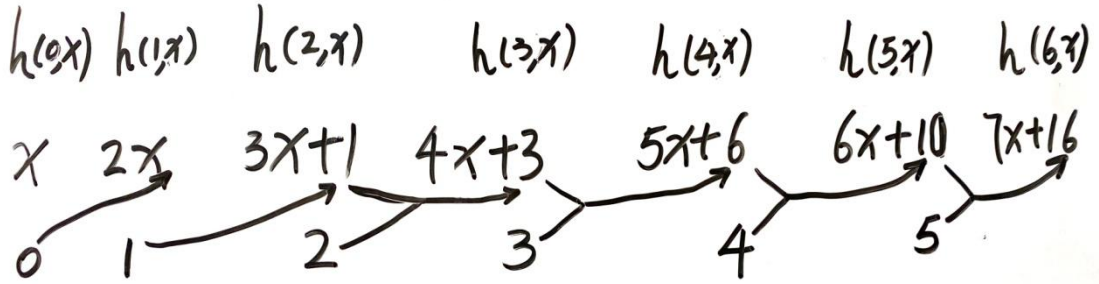
$h(S(n)) = g(h(n), n)$

# 递归与迭代



## 数学上如何构造递归函数

原始递归函数的构造示例




- 已知：f(x)=x

  $g(x_1,x_2,x_3)=x_1+x_2+x_3$, 其中$x,x_1,x_2,x_3$均为自然数

  h(0,x) = f(x)且 h(S(n), x) = g(h(n,x),n,x)

该函数对任一自然数的计算过程为：

h(0,x)=f(x) =x

h(1,x)=h(S(0),x) = g(h(0,x),0,x) = g(f(x),0,x) = f(x) +0+ x =2x

h(2,x)=h(S(1),x) = g(h(1,x),1,x)=g(g(f(x),0,x),1,x)=g(2x, 1, x)=3x+1

h(3,x)=h(S(2),x) =g(h(2,x),2,x)=g(g(h(1, x),1, x),2, x)= g(g(g(h(0,x),0,x),1,x),2,x)

= …= 4x+3

… …

单选题 1分

设置

36

按原始递归的定义，h是由f和g递归地构造出来的。假设已知h(n) = n!，请给出构造h的f和g的函数。正确的是_____。

A f()是常数为1的函数；$g(x_1,x_2) = x_1 * x_2$。

B f()是常数为1的函数；$g(x_1,x_2) = x_1 * (x_2+1)$。

C f()是常数为1的函数；$g(x_1,x_2) = (x_1+1)*(x_2+1)$。

D f()是常数为1的函数；$g(x_1) = n * (x_1)$

提交

# 递归与迭代



## 习题解答

按原始递归的定义，h是由f和g递归地构造出来的。假设已知h(n) = n!，请给出构造h的f和g的函数。正确的是_____。

(A) f()是常数为1的函数；$g(x_1,x_2) = x_1 * x_2$。

(B) f()是常数为1的函数；$g(x_1,x_2) = x_1 * (x_2+1)$

(C) f()是常数为1的函数；$g(x_1,x_2) = (x_1+1)*(x_2+1)$。

(D) f()是常数为1的函数；$g(x_1) = n * (x_1)$

$h(0) = f(\ ) = 1$

$h(n+1)$

$= g(h(n), n)$

A. $h(n+1) = h(n) \times n$

B. $h(n+1) = h(n) \times (n+1)$

C. $h(n+1) = (h(n)+1) \times (n+1)$

D. g 的参数应为2个

# 递归与迭代



## 体验两种不同的递归函数

- Fibonacci数列，无穷数列1，1，2，3，5，8，13，21，34，55，……，

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

递归定义

F(0)=1;

F(1)=1;

F(2)=F(1)+F(0)=2;

F(3)=F(2)+F(1)= 3;

F(4)=F(3)+F(2)= 3+2=5;… …

递推计算/迭代计算/迭代执行

定义是递归的，但执行可以是递归的也可是迭代的

# 递归与迭代



## 体验两种不同的递归函数

**阿克曼递归函数---双递归函数**

$$
\begin{aligned}
A(1,0) &= 2 \\
A(0,m) &= 1 && m \geq 0 \\
A(n,0) &= n+2 && n \geq 2 \\
A(n,m) &= A(A(n-1,m),m-1) && n,m \geq 1
\end{aligned}
$$

$\because A(n,1)=2\times n$

$A(n,2)=A(A(n-1,2),1)=2A(n-1,2)$

$=2\times 2\times A(n-2,2)=\cdots$

$=2^n$

**递归定义**

函数本身是递归的，

函数的变量也是递归的

m=0时， A(n,0)=n+2；

m=1时， A(n,1)=A(**A(n-1,1),0)=A(n-1,1)+2**， 和A(1,1)=A(**A(0,1),0)**=2故A(n,1)=2*n

m=2时， A(n,2)=A(**A(n-1,2),1)=2A(n-1,2)**， 和A(1,2)=A(A(0,2),1)=A(1,1)=2， 故A(n,2)= $2^n$。

m=3时， 类似的可以推出 $\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}}_{n}$

**递归计算/递归执行**

由后向前代入，再由前向后计算

# 递归与迭代



## 体验两种不同的递归函数

**阿克曼递归函数---另一种形式**

$$A(m,n)=\begin{cases} n+1 & 若m=0 \\ A((m-1),1) & 若n=0 \\ A(m-1,A(m,n-1)) & 若m,n>0 \end{cases}$$

**A(1,2) = A(0,A(1,1)) = A(0, A(0,A(1,0))) = A(0, A(0,A(0,1)))=A(0,A(0,2))=A(0,3)=4。**
**A(1,3) =A(0, A(1,2))=A(0. <…代入前式计算过程>)=A(0,4)=4+1=5。**
**…**
**A(1,n) =A(0, A(1,n-1))=A(0. <…代入前式计算过程>)=A(0,n+1)=n+2。**

# 递归与迭代



## 体验两种不同的递归函数

$$A(m,n)=\begin{cases} n+1 & \text{若}m=0 \\ A((m-1),1) & \text{若}n=0 \\ A(m-1,A(m,n-1)) & \text{若}m,n>0 \end{cases}$$

A(2,1)= A(1， A(2， 0))　　--A(2,1)按m,n>0公式代入
= A(1， A(1， 1))　　--A(2,0)按n=0公式代入
=A(1， A(0， A(1， 0)))　　--A(1,1)按m,n>0公式代入
= A(1， A(0， A(0， 1)))　　--A(1,0)按n=0公式代入
= A(1， A(0， 2))　　--A(0,1)按m=0计算
= A(1， 3)　　--A(0,2)按m=0计算
=A(0， A(1， 2))　　--A(1,3)按m,n>0公式代入
= A(0， A(0， A(1， 1)))　　--A(1,2)按m,n>0公式代入
= A(0， A(0， A(0， A(1， 0))))　　--A(1,1)按m,n>0公式代入
= A(0， A(0， A(0， A(0,1)))　　--A(1,0)按n=0公式代入
= A(0， A(0， A(0， 2)))　　--A(0,1)按m=0公式计算
= A(0， A(0， 3)) = A(0， 4) = 5。　　--依次按m=0公式计算

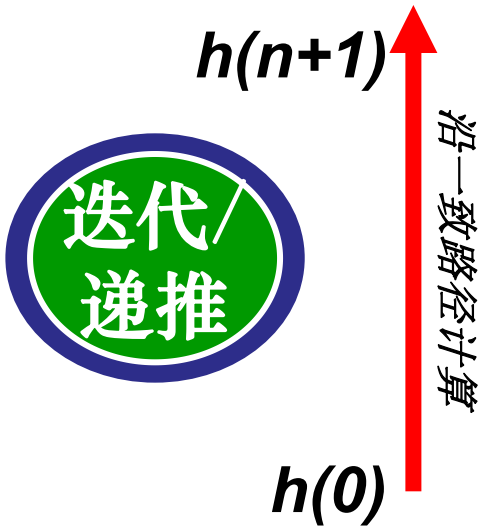
# 递归与迭代



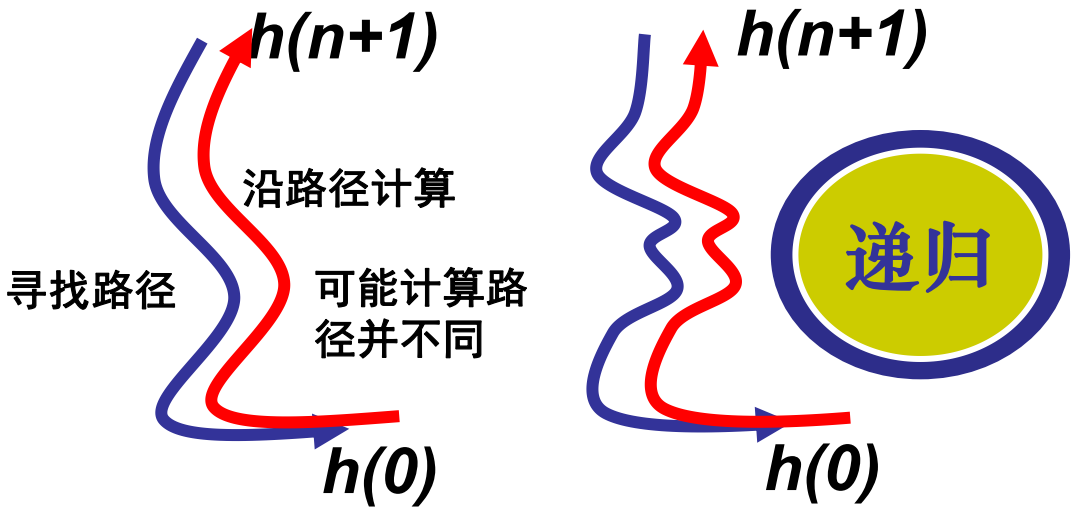
## 递归与迭代的差异

$$h(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ h(n-1)+h(n-2) & n>1\end{cases}$$

$$h(m,n)=\begin{cases}n+1 & \text{若}m=0\\ h((m-1),1) & \text{若}n=0\\ h(m-1,h(m,n-1)) & \text{若}m,n>0\end{cases}$$




在前次结果基础上进行计算
可采用循环结构实现




通常，只能由函数结构实现

单选题 1分

设置

43

关于递归定义的函数，下列说法正确的是_____。

A 递归定义的函数一定是“递归计算”的；

B 递归定义的函数一定是“迭代计算”的；

C 有些递归定义的函数可以“迭代计算”，有些递归定义的函数则必须“递归计算”；

D 凡是可以“迭代计算”的函数，一定可以“递归计算”，凡是可以“递归计算”的函数，也一定可以“迭代计算”。

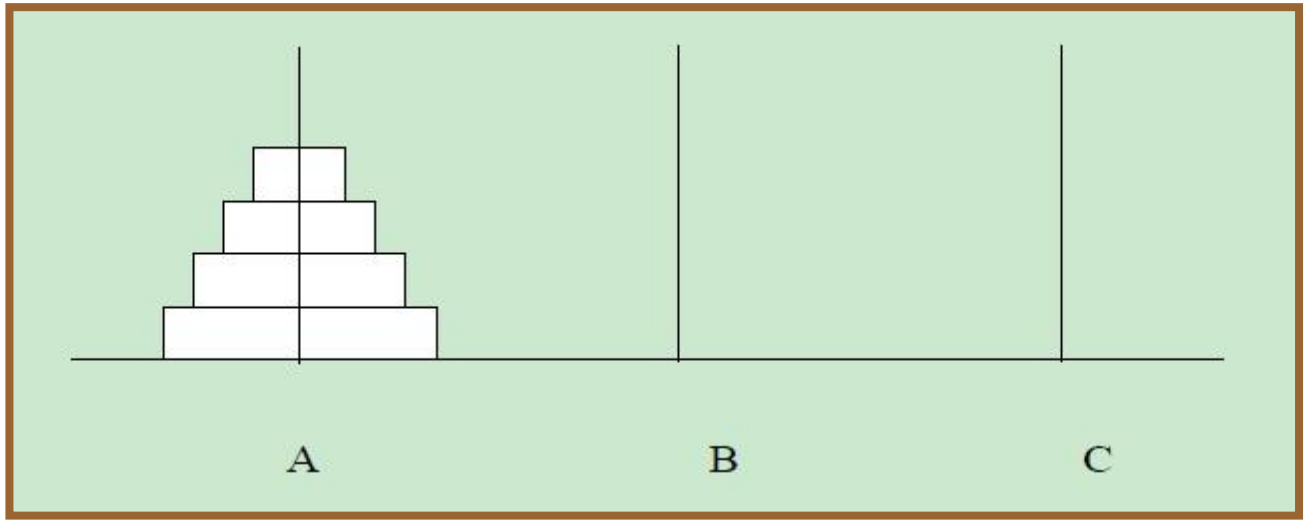
提交

# 递归算法/程序示例一



## 汉诺塔问题

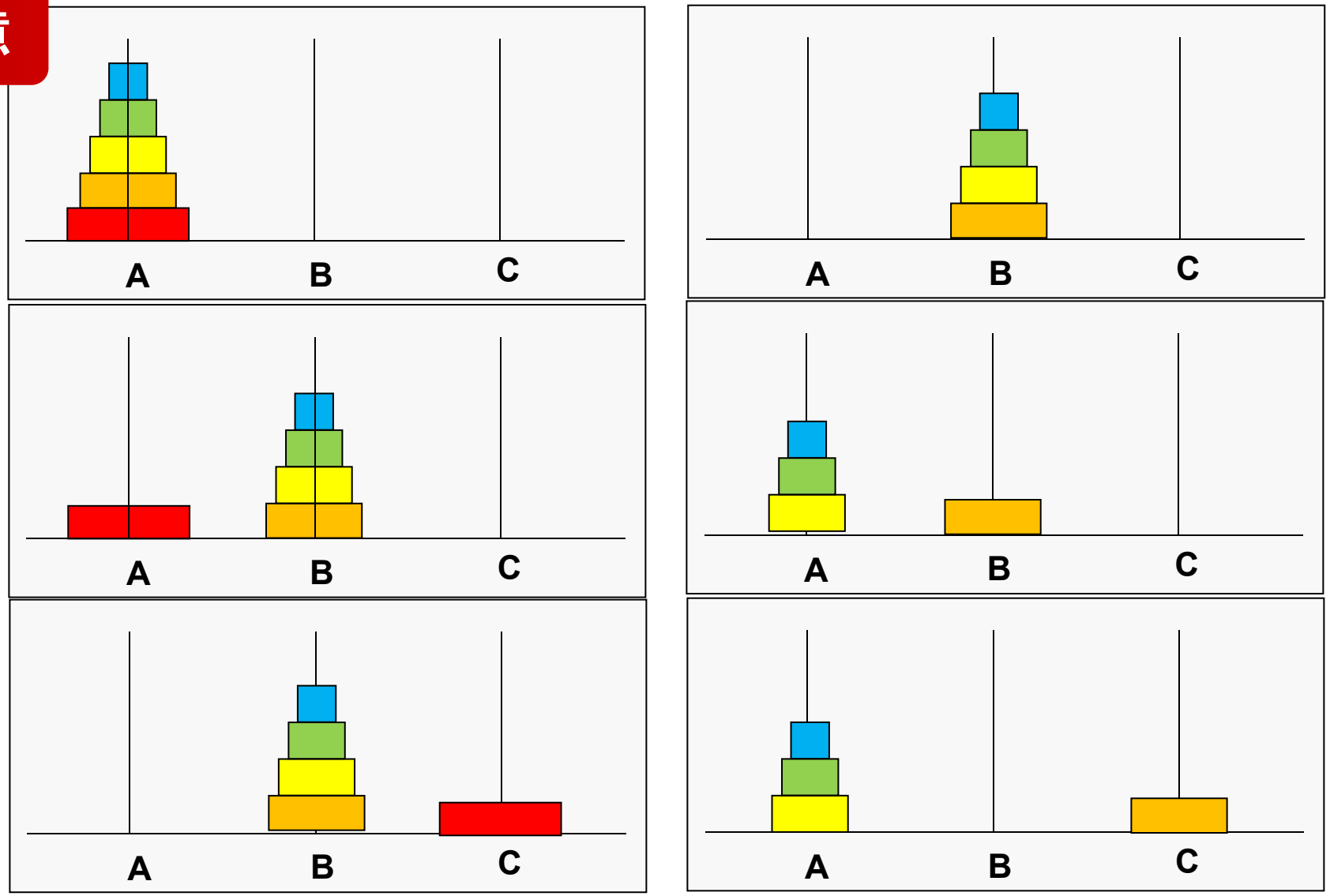
**梵天塔(汉诺塔)问题**：有三根柱子，梵天将64个直径大小不一的金盘子按照从大到小的顺序依次套放在第一根柱子上形成一座金塔，要求每次只能移动一个盘子，盘子只能在三根柱子上来回移动不能放在他处，在移动过程中三根柱子上的盘子必须始终保持大盘在下小盘在上。



# 递归算法/程序示例一



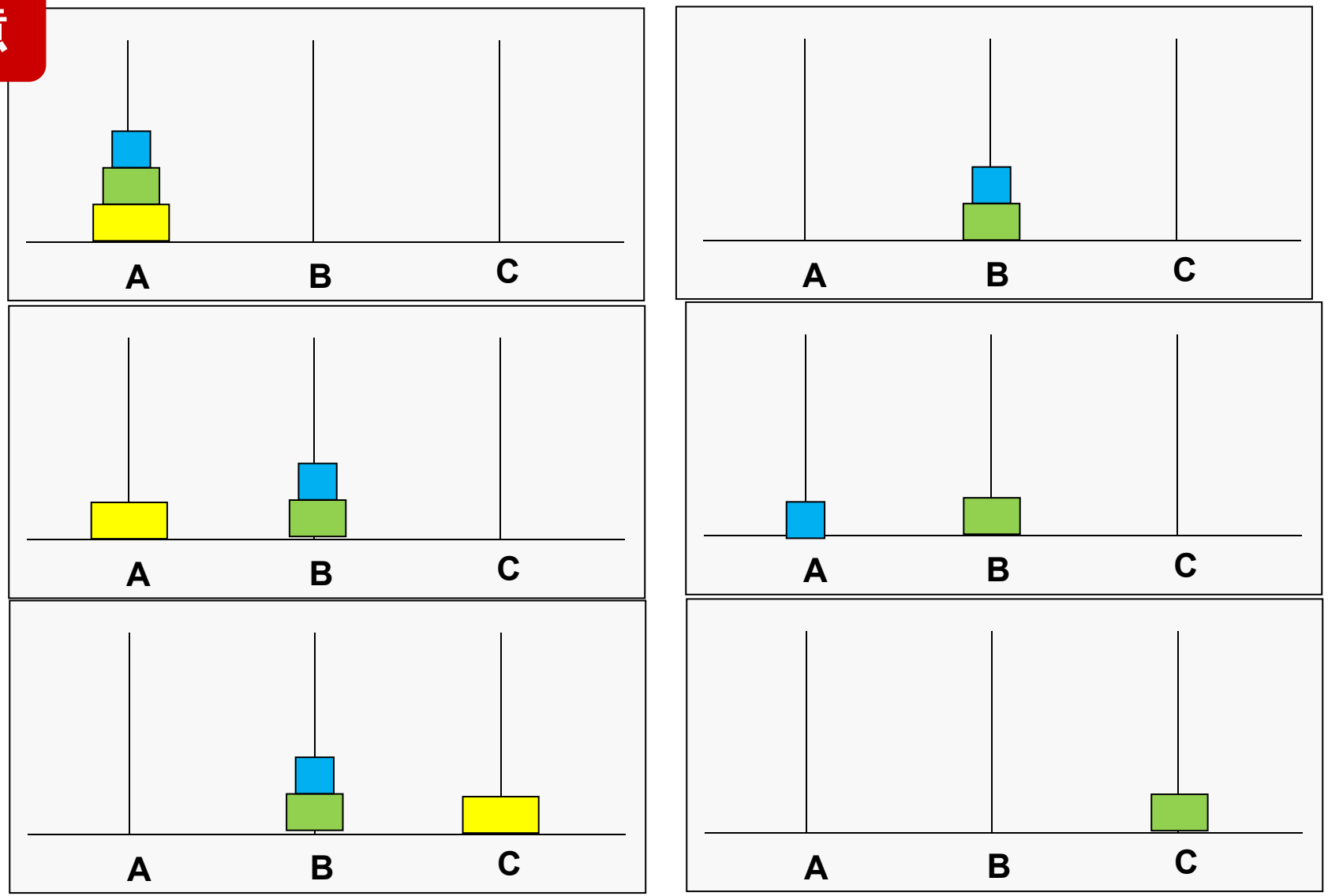
## 汉诺塔问题求解示意



# 递归算法/程序示例一



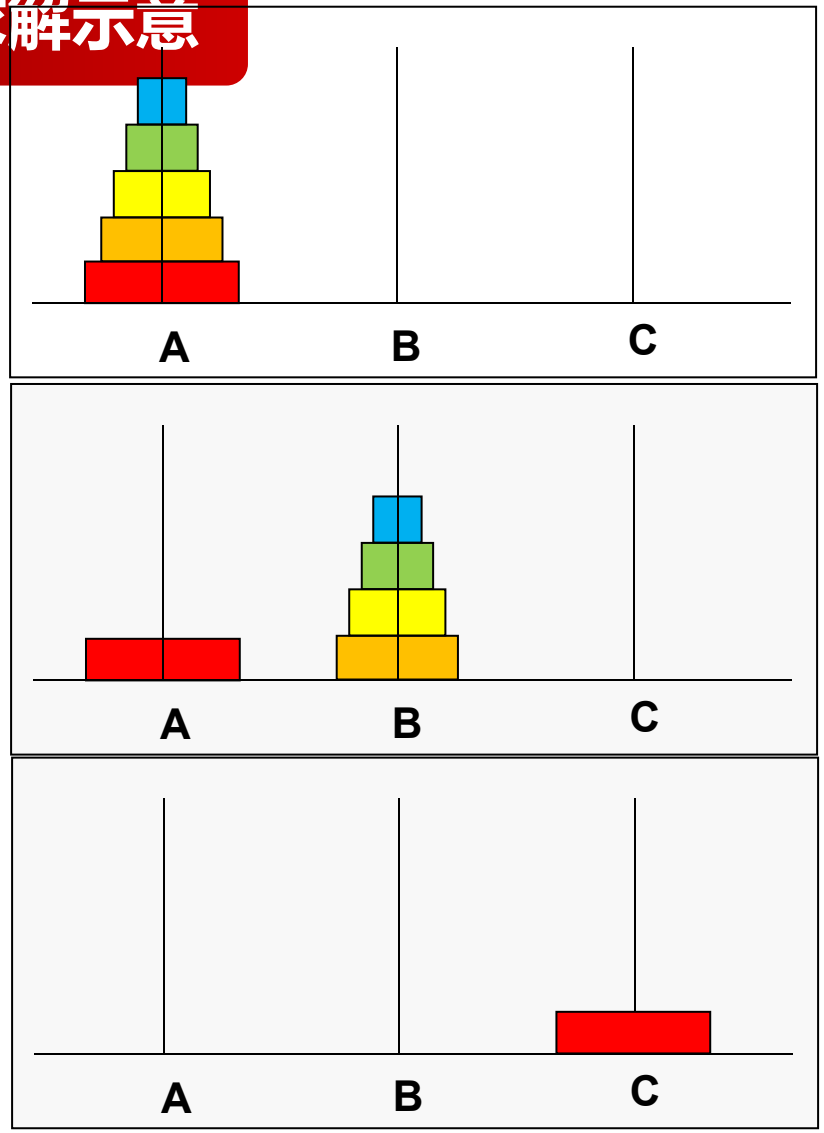
## 汉诺塔问题求解示意



# 递归算法/程序示例一



## 汉诺塔问题求解示意




```
Main()
{ //假设有5个盘子的汉诺塔
    Hanoi(5, “A”, “B”, “C”);
}

int Hanoi (int N, int X, int Y, int Z)
{ //该函数是将N个盘子从X柱，以Y柱做中转，移动到Z柱上
  If  N > 1 Then
  {
    //先把n-1个盘子从X放到Y上(以Z做中转)
        Hanoi (N - 1, X, Z, Y);
    //然后把X上最下面的盘子放到Z上
        Printf( “Dish: %d:%d→%d” , N, X, Z);
    //接着把n-1个盘子从Y上放到Z上(以X做中转)
        Hanoi (N - 1, Y, X, Z);
  }
  Else
  { //只有一个盘子时，直接把它从X放到Z上
        Printf( “Dish: %d:%d→%d” , N, X, Z);
  }
}
```

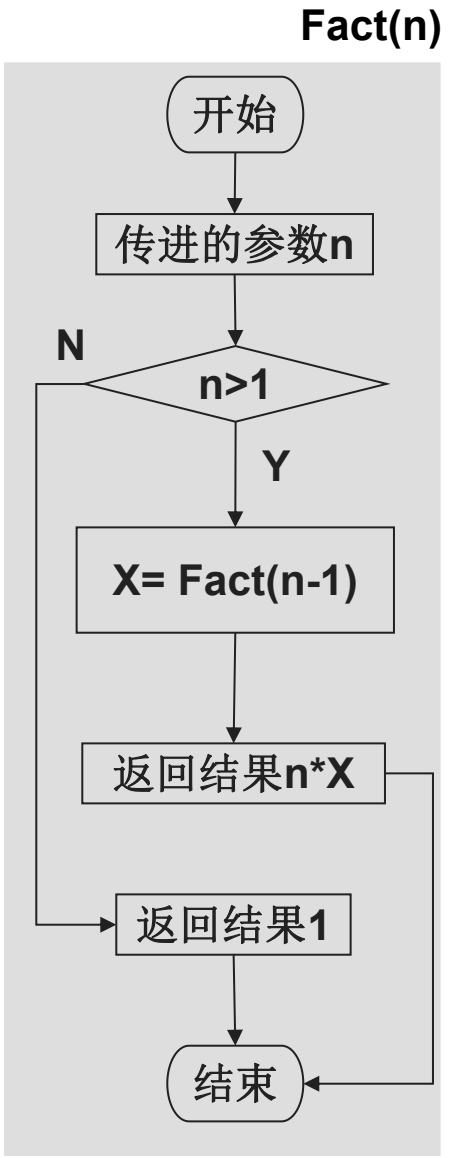
# 递归算法/程序示例二



## 递归法求n!的算法或程序

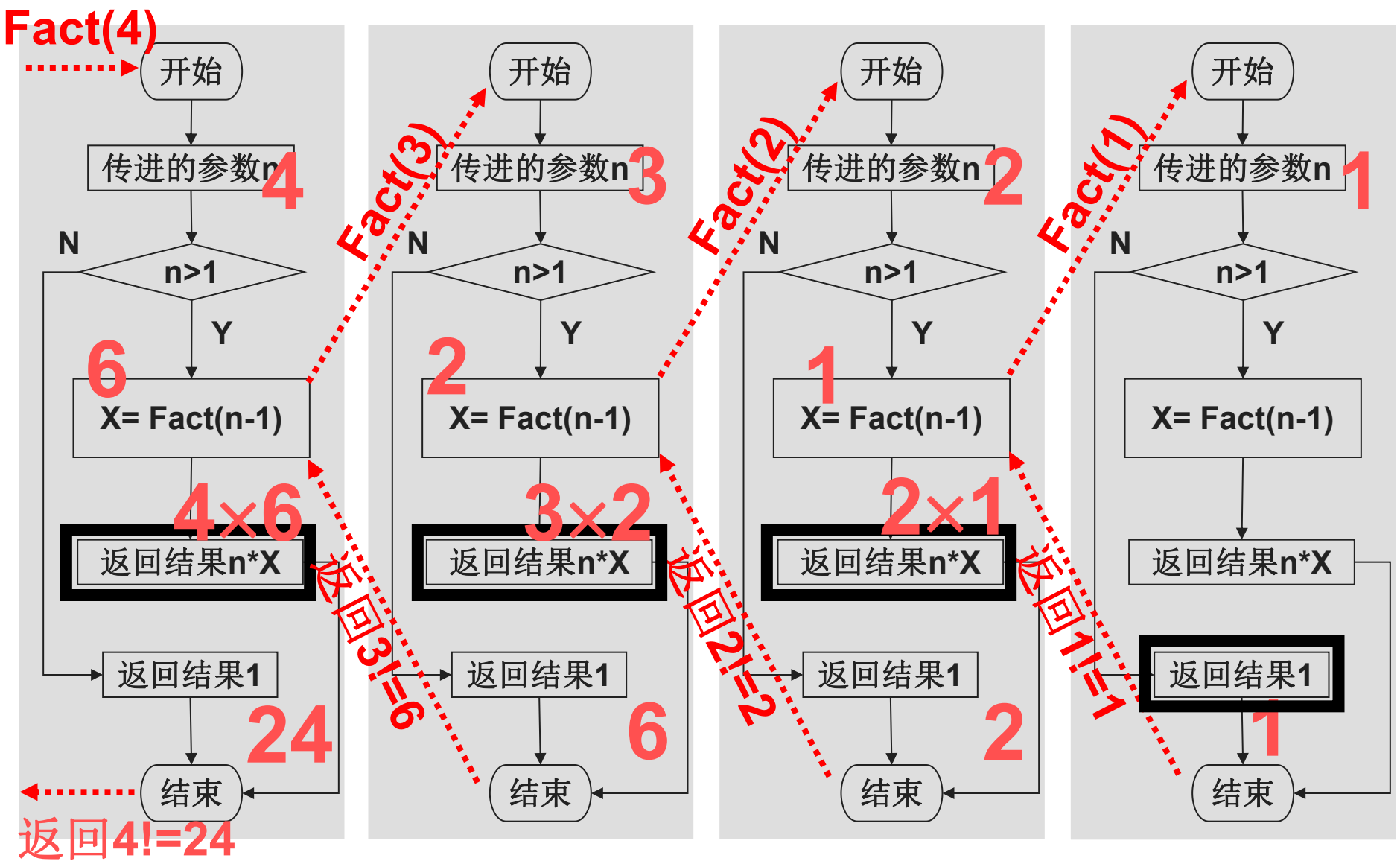
$$n!=\begin{cases}1 & \text{当} n\le 1 \text{时}\\ n\times(n-1)! & \text{当} n>1 \text{时}\end{cases}$$

```
long int Fact(int n)
{    long int x;
     If (n > 1)
     { x = Fact(n-1);
  /*递归调用*/
        return n*x;   }
     else return 1;
  /*递归基础*/
}
```



# 递归算法/程序示例二




Fact(4)
开始
传进的参数n
4
N
n>1
Y
6
X= Fact(n-1)
4×6
返回结果n*X
返回结果1
24
结束
返回4!=24
Fact(3)
返回3!=6
开始
传进的参数n
3
N
n>1
Y
2
X= Fact(n-1)
3×2
返回结果n*X
返回结果1
6
结束
Fact(2)
返回2!=2
开始
传进的参数n
2
N
n>1
Y
1
X= Fact(n-1)
2×1
返回结果n*X
返回结果1
2
结束
Fact(1)
返回1!=1
开始
传进的参数n
1
N
n>1
Y
X= Fact(n-1)
返回结果n*X
返回结果1
1
结束

# 递归算法/程序示例二



## 迭代法求n!的算法或程序

$$n!=\begin{cases}1 & \text{当} n\le 1\text{时}\\ 1\times 2\times\ldots(n-1)\times n & \text{当} n>1\text{时}\end{cases}$$

```
long int Fact(int n)
{   int  counter;
    long product=1;
    for  counter=1 to n step 1
    { product = product * counter;   }
  /*迭代*/
return product;
}
```

| Counter | Product |
|---|---|
| | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |

# 组合、抽象与构造的程序思维小结



## 程序思维小结