大学计算机-计算思维导论
第三章 问题求解框架

哈尔滨工业大学　（深圳）
计算机学院

主要内容

1.传统程序构成要素

常量、变量、表达式语句、函数

数据结构

程序构造及表达

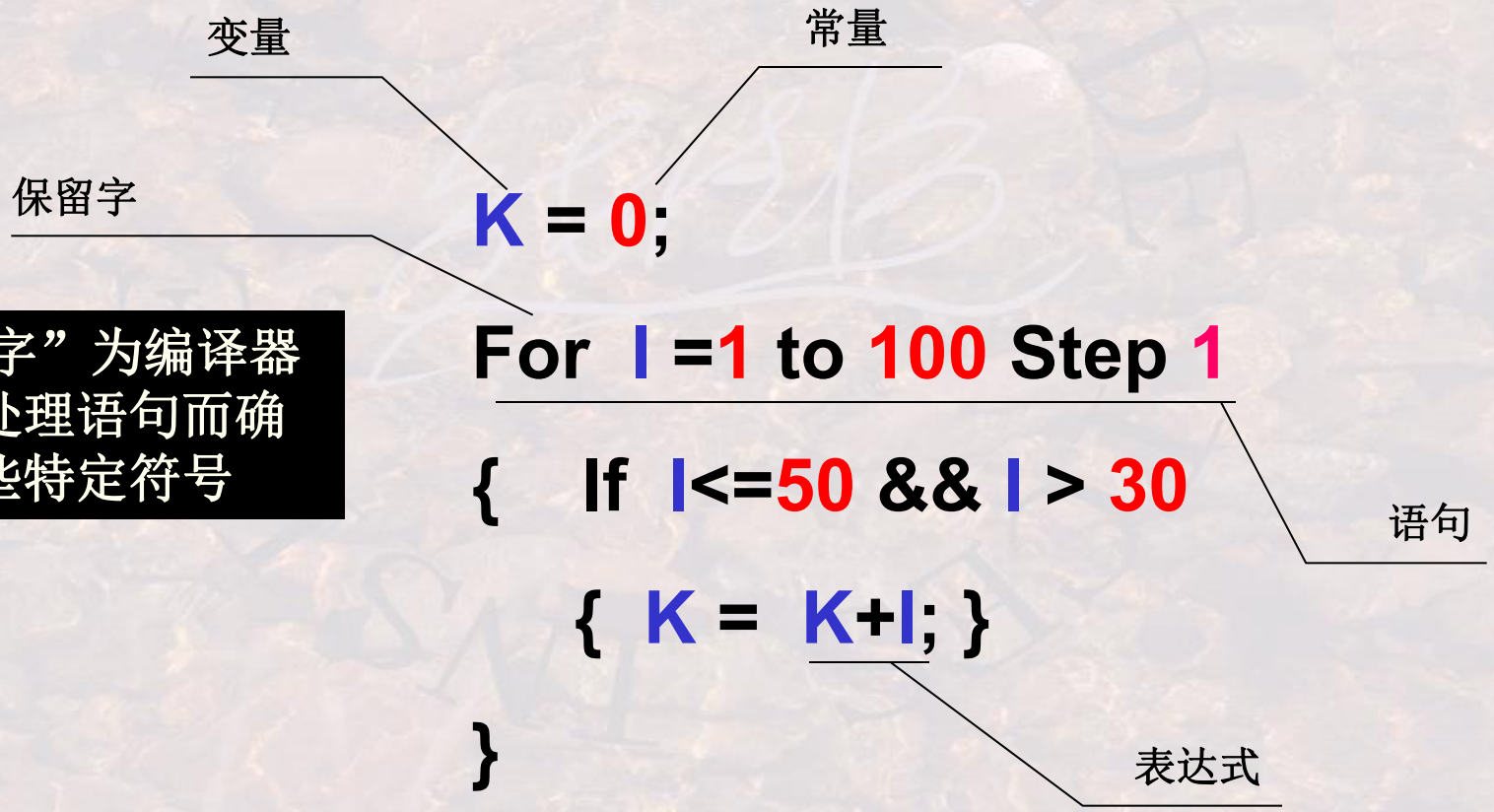
2.现代程序的基本构成要素

类、对象、消息

3.1 传统程序的基本构成要素

（1） 计算机语言程序的基本构成要素有哪些？

认识计算机语言程序



“保留字”为编译器识别和处理语句而确定的一些特定符号

（2） 你能够书写三种形式的表达式吗?

常量、变量与表达式

A1 + (B2 – x1 + 76) * 3

(+ A1 (* (+ (- B2 x1) 76) 3)

◆算术表达式示例。算术表达式的结果是一数值；

A1 + (B2 – x1 + 76) * 3

(B2 + yy4) / L3 – xx3

◆关系表达式示例。关系表达式的计算结果是逻辑“真”或“假”；

Grade < 90

Grade >= 70

N4 < A1 + B2 + 20 //注：A1+B2+20为算术表达式，计算完后再与N4的值进行比较

◆逻辑表达式示例。逻辑表达式的计算结果是逻辑“真”或“假”；

(x1 >= A1) && (B2 <> y2)

◆将表达式的计算结果赋值给一变量：赋值语句

M = X>Y+50;

M = (X>Y) AND (X<Y);

K = K + (5 * K);

语句与程序控制

◆顺序结构

```
G5 = 1;
G6 = 2;
G7 = 3;
G8 = 4;
G9 = 5;
G9 = G9 + G8;
G9 = G9 + G7;
G9 = G9 + G6;
G9 = G9 + G5;
```

程序执行示例

```
G5 = 1;
G6 = 2;
G7 = 3;
G8 = 4;
G9 = 5;
G9 = G9 + G8;
G9 = G9 + G7;
G9 = G9 + G6;
G9 = G9 + G5;
```

单选题 1分

设置

已知A=40；B=30；C=100；计算表达式(A > B) and (B < C)的值，结果为

A 100

B 30

C 真

D 假

提交

语句与程序控制

◆分支结构

IF 条件表达式 {
(条件为真时运行的)程序语句序列1 }
ELSE {
(条件为假时运行的)程序语句序列2 }

```
If  D1>D2
{    D1=D1-5;   }
Else
{   D1=D1+10;  }
```

```
Y = 50;
Z = 80;
X = 30;
X = Z + Y;
If  Y > Z   {
      X = X - Y;  }
Else {
      X= X - Z;   }
X = X + Y;
If  X > Z  {  X = Y; }
X = X - Z;
If  X>Y
{ X = X - Y;   }
```

语句与程序控制


```
Y = 50;
Z = 80;
X = 30;
X = Z + Y;
If  Y > Z   {
      X = X - Y;  }
Else {
      X= X - Z;   }
X = X + Y;
If  X > Z  {  X = Y; }
X = X - Z;
If  X>Y
{ X = X - Y;   }
```

语句与程序控制

◆循环结构(有界循环结构)

For (计数器变量 = 起始值 To 结束值 [增量表达式])
{ 循环体的程序语句序列 }
Next [计数器变量]

```
Sum=0;
For  I = 1 to 5 Step 1
{  Sum = Sum + I;  }
Next I
//继续其他语句
```

```
Sum=0;
For  I =1 to 10000 Step 2
{ Sum = Sum + I; }
Next I
```

Sum [00]

I [6]

语句与程序控制

◆循环结构(条件循环结构)

Do

{ 循环体的程序语句序列 }

While (条件表达式);


```
X=1;
Y=2;
Sum=0;
Do {
    Sum = X+Y;
    X=X+1;
    Y=Y+1;
} While (Sum<=10)
//其他语句
```

语句与程序控制

◆循环结构(条件循环结构)

Do

{ 循环体的程序语句序列 }

While (条件表达式);


```
X=1;
Y=2;
Sum=0;
Do {
    Sum = X+Y;
    X=X+1;
    Y=Y+1;
} While (Sum<0)
//其他语句
```

语句与程序控制

◆循环结构(条件循环结构)

While (条件表达式)

Do { 循环体的程序语句序列 }


```
X=1;
Y=2;
Sum=0;
While (Sum<0)
    Do {
        Sum = X+Y;
        X=X+1;
        Y=Y+1;
        }
<其他语句>
```

单选题 1分

设置

```
K = 0;
I = 2;
While (I<=8)
{   K = K + I;
I = I + 2;  }
```

该程序执行完成后，K的值为_____。

A 35

B 20

C 36

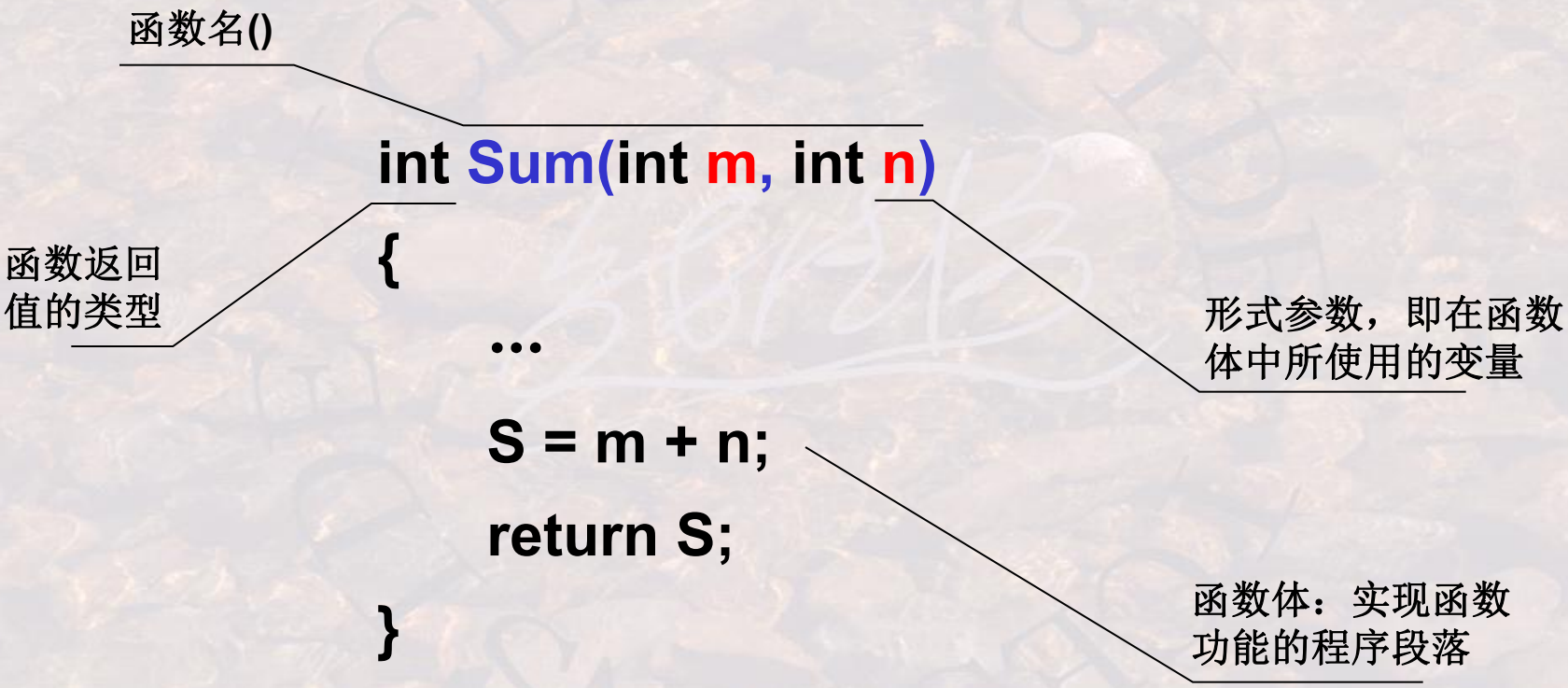
D 12

提交

3.1.3 函数与函数调用

（1） 函数是很重要的程序构造手段，你知道吗？

函数

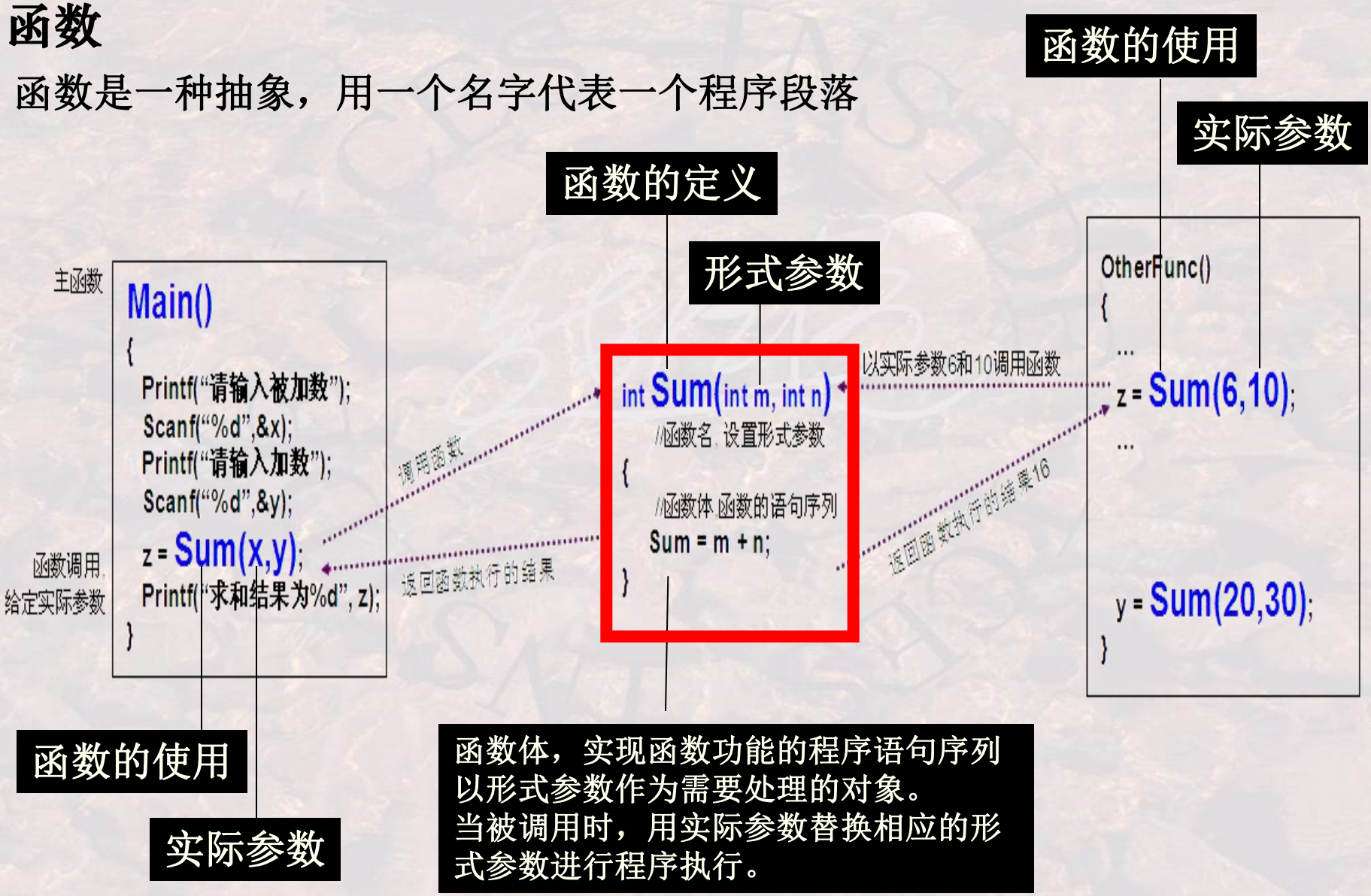


数学上的函数只是一个符号表达，而计算机程序中的函数则是一段可以执行的程序

（2）你知道函数是一种抽象吗？

函数

函数是一种抽象，用一个名字代表一个程序段落



（3） 你知道计算机语言或操作系统提供哪些函数吗?

系统提供的可以使用的函数类别

- **数学运算函数**，如三角函数、指数与对数函数、开方函数等；例如sin(α)，Log(x)等；
- **数据转换函数**，如字母大小写变换、数值型数字和字符型数字相互转换等；
- **字符串操作函数**，如取子串、计算字符串长度等；例如，Len("abcd")；
- **输入输出函数**，如输入输出数值、字符、字符串等；例如，Printf(…),Scanf(…)等；
- **文件操作函数**，如文件的打开、读取、写入、关闭等；
- **其它函数**，如取系统日期、绘制图形等。

3.1.4. 常量/变量、数据存储与数据结构
(1)符号化常量与变量的区别

符号化常量：命名计算对象和程序中使用符号化常量及计算中以计算对象替换符号化常量

```
define pi 3.14159
define STR1 “Hello！”
或
Const pi = 3.14159
Const STR1 = “Hello！”

X=30*pi
```

变量：用若干个存储单元来表示，变量名被编译成了存储单元的地址

变量及其存储

◆变量与存储单元

	变量名	变量值
Int X=23;	*x*	00000000 00010111
Char Y= 'AB';	*y*	01000001 01000010
Char Z= 'ABCD';	*Z*	01000001 01000010
		01000011 01000100
		00000000

存储地址	存储内容
00000000 00000001	00000000 00010111
00000000 00000010	01000001 01000010
00000000 00000011	01000001 01000010
00000000 00000100	01000011 01000100
00000000 00000101	00000000
00000000 00000110	

3.1.4. 常量/变量、数据存储与数据结构
（2）变量及其变量类型与存储单元

◆“变量”与“变量类型”及其存储

用名字表示的存储地址，即变量名	存储地址	存储内容(即变量值)
Mark long int	00000000 00000000 00000000 00000001 00000000 00000010 00000000 00000011	(注：可通过赋值发生改变)
Sum int	00000000 00000100 00000000 00000101	(注：可通过赋值发生改变)
Distance int	00000000 00000110 00000000 00000111	(注：可通过赋值发生改变)

变量及其存储

◆ “变量”与“指针变量”

String *P=”ABCD”;	P
String v = “ABCD”;	v *p

存储地址	存储内容
00000000 00000000	00000100 00001000
00000000 00000001	00000000 00000100
00000000 00000010	00001100 00001010
00000000 00000011	00000000 00000000
00000000 00000100	01000001 01000010
00000000 00000101	01000011 01000100
00000000 00000110	00000000 00000000
00000000 00000111	01000001 01000010
00000000 00001000	01000011 01000100
00000000 00001001	00000000 00000000

3.1.4. 常量/变量、数据存储与数据结构
(3)什么是数据结构?

数据结构是数据的逻辑结构、存储结构及其运算的总称，它提供了问题求解/算法的数据操纵机制。

逻辑结构：数据之间的关系

存储结构：数据在存储器中的存储方式，顺序和链式

基本运算：建立，插入，删除等

典型数据结构：向量，列表，数组，树，图等。

多元素变量及其存储

◆向量或列表是有序数据的集合型变量，向量中的每一个元素都属于同一个数据类型，用一个统一的向量名和下标来唯一的确定向量中的元素。在程序设计语言中，又称为数组。

◆向量名通常表示该向量的起始存储地址，而向量下标表示所指向元素相对于起始存储地址的偏移位置。

向量实例

82	Mark[0]
95	Mark[1]
100	Mark[2]
60	Mark[3]
80	Mark[4]

编写求上述数组中值的平均值的程序

```
n = 4;
Sum=0;
For  J =0 to n Step 1
{    Sum = Sum + mark[ J ];
        }
Next  J
Avg = Sum/(n+1);
```

向量存储实例

用变量名和元素位置共同表示存储地址，即向量		存储地址	存储内容(即变量值)
Mark	[0]	00000000 00000000 00000000 00000001	(注：82 的 4 字节二进制数可通过赋值发生改变)
	[1]	00000000 00000010 00000000 00000011	(注：95 的 4 字节二进制数可通过赋值发生改变)
	[2]	00000000 00000100 00000000 00000101	(注：100 的 4 字节二进制数可通过赋值发生改变)
	[3]	00000000 00000110 00000000 00000111	(注：60 的 4 字节二进制数可通过赋值发生改变)
	[4]	00000000 00001000 00000000 00001001	(注：80 的 4 字节二进制数可通过赋值发生改变)

多元素变量使得程序可通过下标来操作多元素变量中的每一个元素

多元素变量及其存储

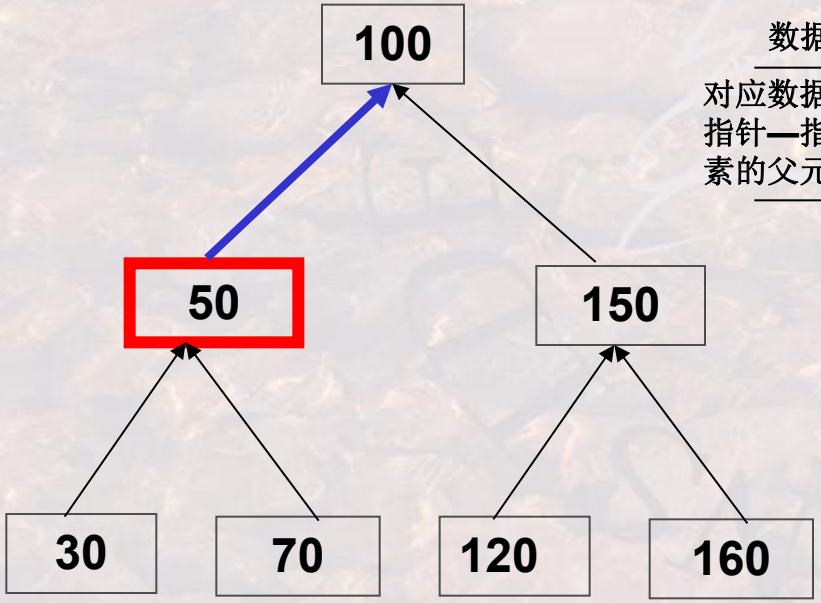
◆矩阵或表是按行按列组织数据的集合型变量，通常是一个二维向量，可表示为如M[2,3]或M[2][3]形式，即用符号名加两个下标来唯一确定表中的一个元素，前一下标为行序号，后一下标为列序号。系统会自动将其转换为对应的存储地址，找到相应的存储单元。在程序设计语言中，矩阵或表是一个多维数组变量。


```
Sum=0;
For  I =1 to 4 Step 1
{   For  J =1 to 4 Step 1
    { Sum = Sum + M[I][J];  }
    Next  J
}
Next  I
Avg = Sum/16;
```

逻辑上是二维的按行、列下标来操作一个元素，如M[2,3]或M[2][3]；物理上仍旧是一维存储的，由“表起始地址+ (行下标-1)*列数/行+列下标”。这种转换可由系统自动完成，程序中只需按下标操作即可，即如M[2][3]

典型的数据结构---“树”

“树”的存储结构：一个存储单元存储“数据元素”；一个存储单元存储“指针”，指示数据元素之间的逻辑关系



数据的逻辑结构

数据的存储结构

数据元素

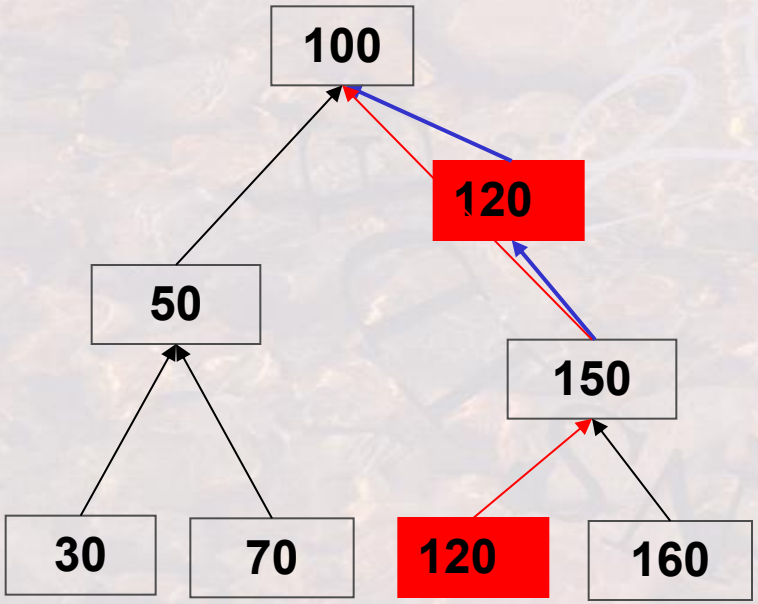
对应数据元素的指针—指向该元素的父元素

存储地址	存储内容(即变量值)	注释
00000000 00000001	00000000 01100100	第 1 个数据元素 100
00000000 00000010	00000000 00000000	第 1 个数据元素的指针
00000000 00000011	00000000 00110010	第 2 个数据元素 50
00000000 00000100	00000000 00000001	第 2 个数据元素的指针
00000000 00000101	00000000 10100110	第 3 个数据元素 150
00000000 00000110	00000000 00000001	第 3 个数据元素的指针
00000000 00000111	00000000 00011110	第 4 个数据元素 30
00000000 00001000	00000000 00000011	第 4 个数据元素的指针
00000000 00001001	00000000 01000110	第 5 个数据元素 70
00000000 00001010	00000000 00000011	第 5 个数据元素的指针
00000000 00001011	00000000 01111000	第 6 个数据元素 120
00000000 00001100	00000000 00000101	第 6 个数据元素的指针
00000000 00001101	00000000 10100000	第 7 个数据元素 160
00000000 00001110	00000000 00000101	第 7 个数据元素的指针
00000000 00001111		

3.1.4. 常量/变量、数据存储与数据结构

（3）什么是数据结构？：树

存储结构中，通过指针的变化，不改变数据元素的存储，但却改变了数据元素之间的逻辑关系



数据的逻辑结构

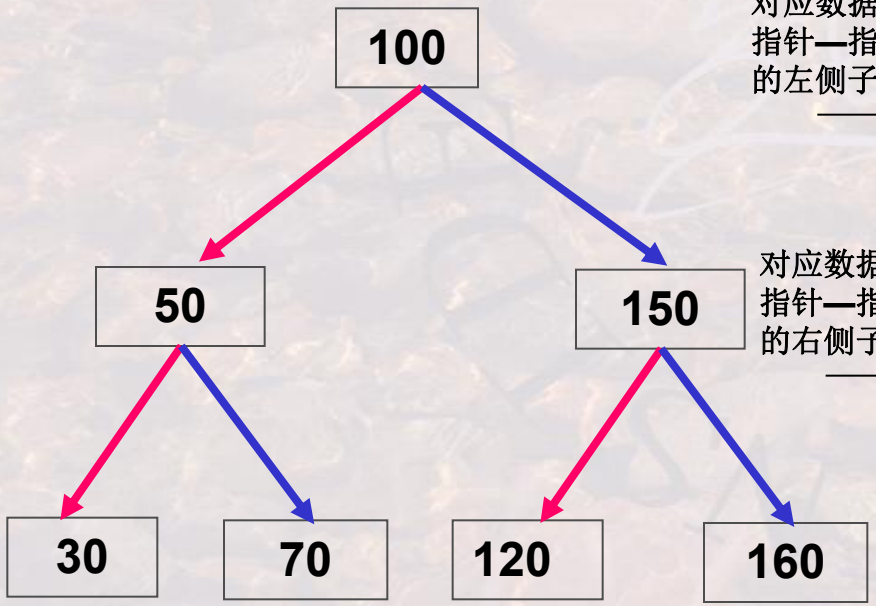
数据的存储结构

存储地址	存储内容(即变量值)	注释
00000000 00000001	00000000 01100100	第 1 个数据元素 100
00000000 00000010	00000000 00000000	第 1 个数据元素的指针
00000000 00000011	00000000 00110010	第 2 个数据元素 50
00000000 00000100	00000000 00000001	第 2 个数据元素的指针
00000000 00000101	00000000 10100110	第 3 个数据元素 150
00000000 00000110	00000000 00001011	第 3 个数据元素的指针
00000000 00000111	00000000 00011110	第 4 个数据元素 30
00000000 00001000	00000000 00000011	第 4 个数据元素的指针
00000000 00001001	00000000 01000110	第 5 个数据元素 70
00000000 00001010	00000000 00000011	第 5 个数据元素的指针
00000000 00001011	00000000 01111000	第 6 个数据元素 120
00000000 00001100	00000000 00000001	第 6 个数据元素的指针
00000000 00001101	00000000 10100000	第 7 个数据元素 160
00000000 00001110	00000000 00000101	第 7 个数据元素的指针
00000000 00001111		

3.1.4. 常量/变量、数据存储与数据结构

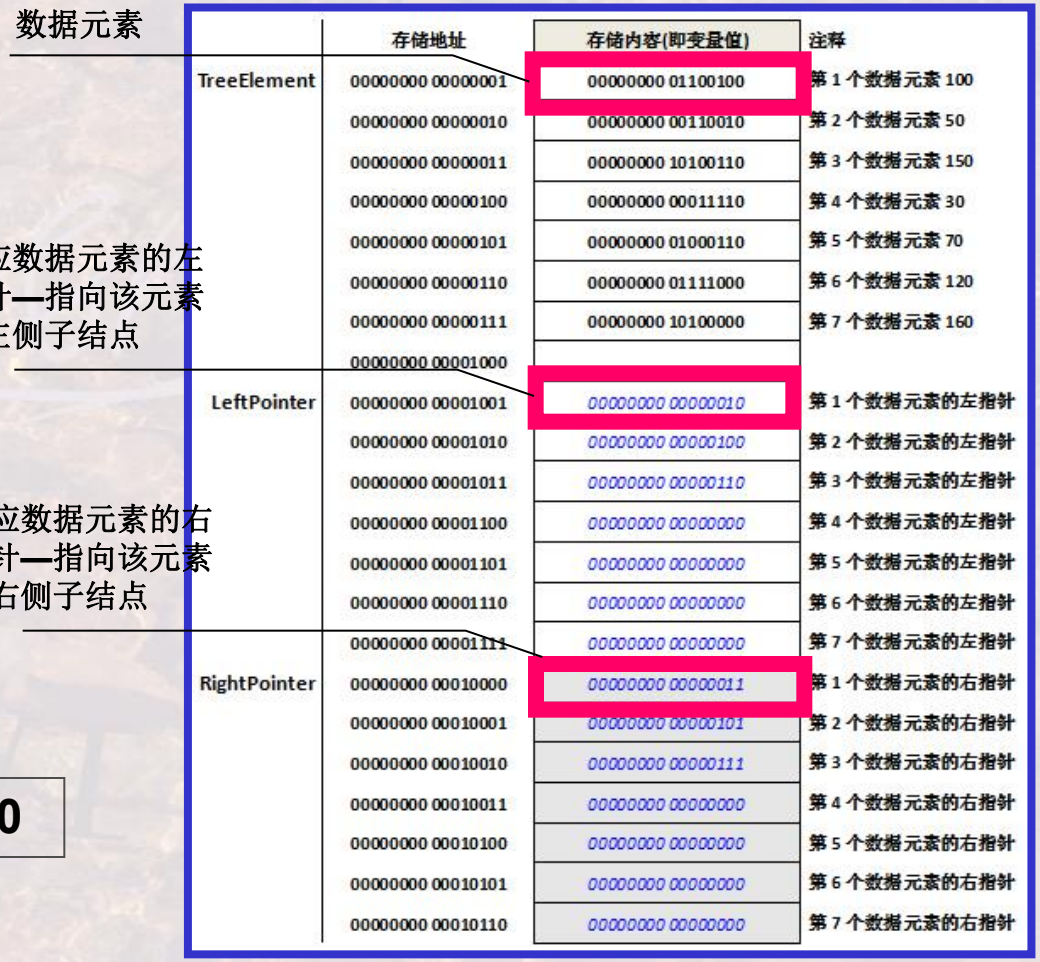
（3）什么是数据结构？：树：同样的逻辑结构可以有不同的存储结构吗？

“树”的另一种**存储结构**，用两个指针表达数据之间的逻辑关系，一个指向其左数据元素，一个指向其右数据元素。



数据的逻辑结构

数据的存储结构



数据结构不同，数据之间的操作方法也是不同的

单选题 1分

设置

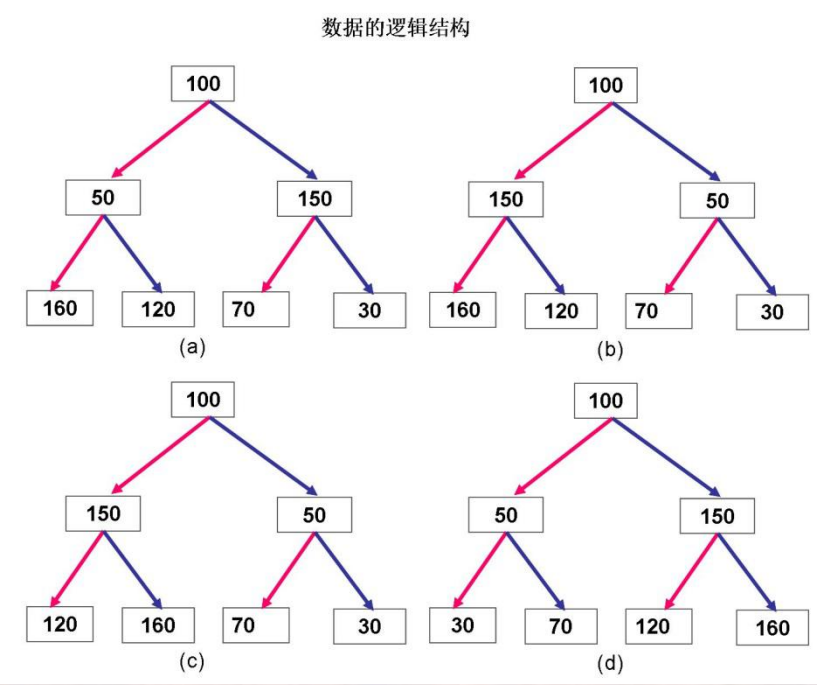
图表示的数据的逻辑关系，正确是

A a

B b

C c

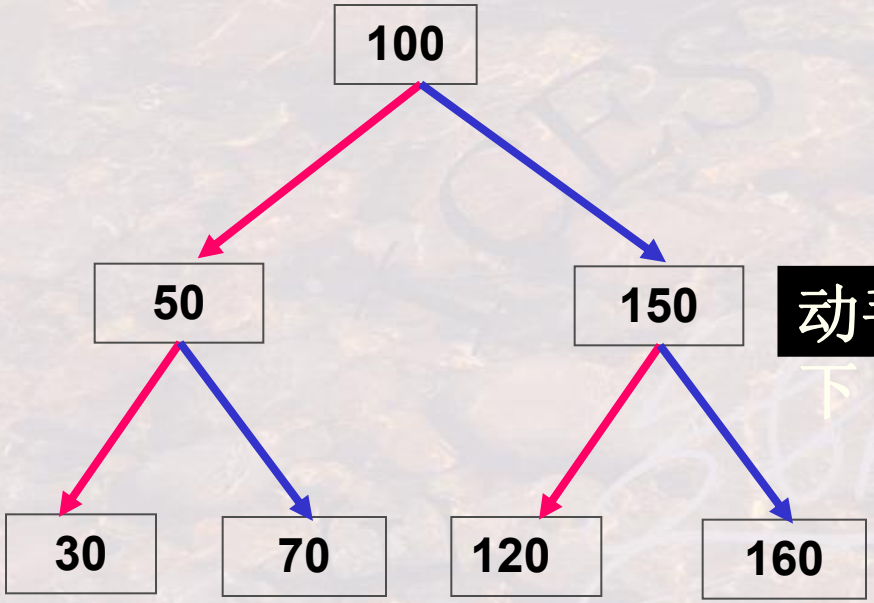
D d



	存储地址	存储内容(即变量值)	注释
TreeElement	00000000 00000001	00000000 01100100	第 1 个数据元素 100
	00000000 00000010	00000000 00110010	第 2 个数据元素 50
	00000000 00000011	00000000 10100110	第 3 个数据元素 150
	00000000 00000100	00000000 00011110	第 4 个数据元素 30
	00000000 00000101	00000000 01000110	第 5 个数据元素 70
	00000000 00000110	00000000 01111000	第 6 个数据元素 120
	00000000 00000111	00000000 10100000	第 7 个数据元素 160
	00000000 00001000		
	00000000 00001001		
LeftPointer	00000000 00001010	*00000000 00000010*	第 1 个数据元素的左指针
	00000000 00001011	*00000000 00000100*	第 2 个数据元素的左指针
	00000000 00001100	*00000000 00000110*	第 3 个数据元素的左指针
	00000000 00001101	*00000000 00000000*	第 4 个数据元素的左指针
	00000000 00001110	*00000000 00000000*	第 5 个数据元素的左指针
	00000000 00001111	*00000000 00000000*	第 6 个数据元素的左指针
	00000000 00010000	*00000000 00000000*	第 7 个数据元素的左指针
	00000000 00010001		
	00000000 00010010		
RightPointer	00000000 00010011	*00000000 00000011*	第 1 个数据元素的右指针
	00000000 00010100	*00000000 00000101*	第 2 个数据元素的右指针
	00000000 00010101	*00000000 00000111*	第 3 个数据元素的右指针
	00000000 00010110	*00000000 00000000*	第 4 个数据元素的右指针
	00000000 00010111	*00000000 00000000*	第 5 个数据元素的右指针
	00000000 00011000	*00000000 00000000*	第 6 个数据元素的右指针
	00000000 00011001	*00000000 00000000*	第 7 个数据元素的右指针
	00000000 00011010		
	00000000 00011011		

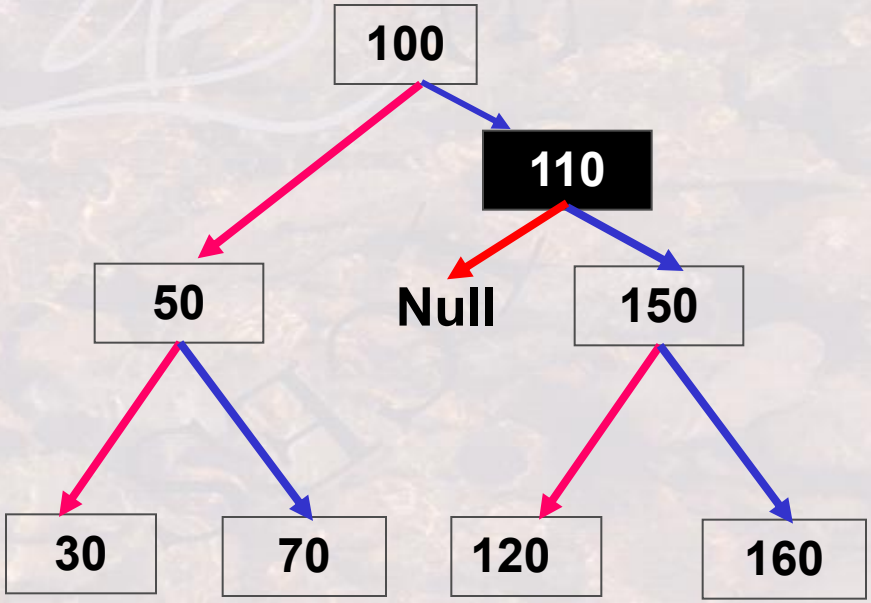
3.1.4. 常量/变量、数据存储与数据结构

（3）什么是数据结构？：树：同样的逻辑结构可以有不同的存储结构吗？



数据的逻辑结构

动手练习一



数据的逻辑结构

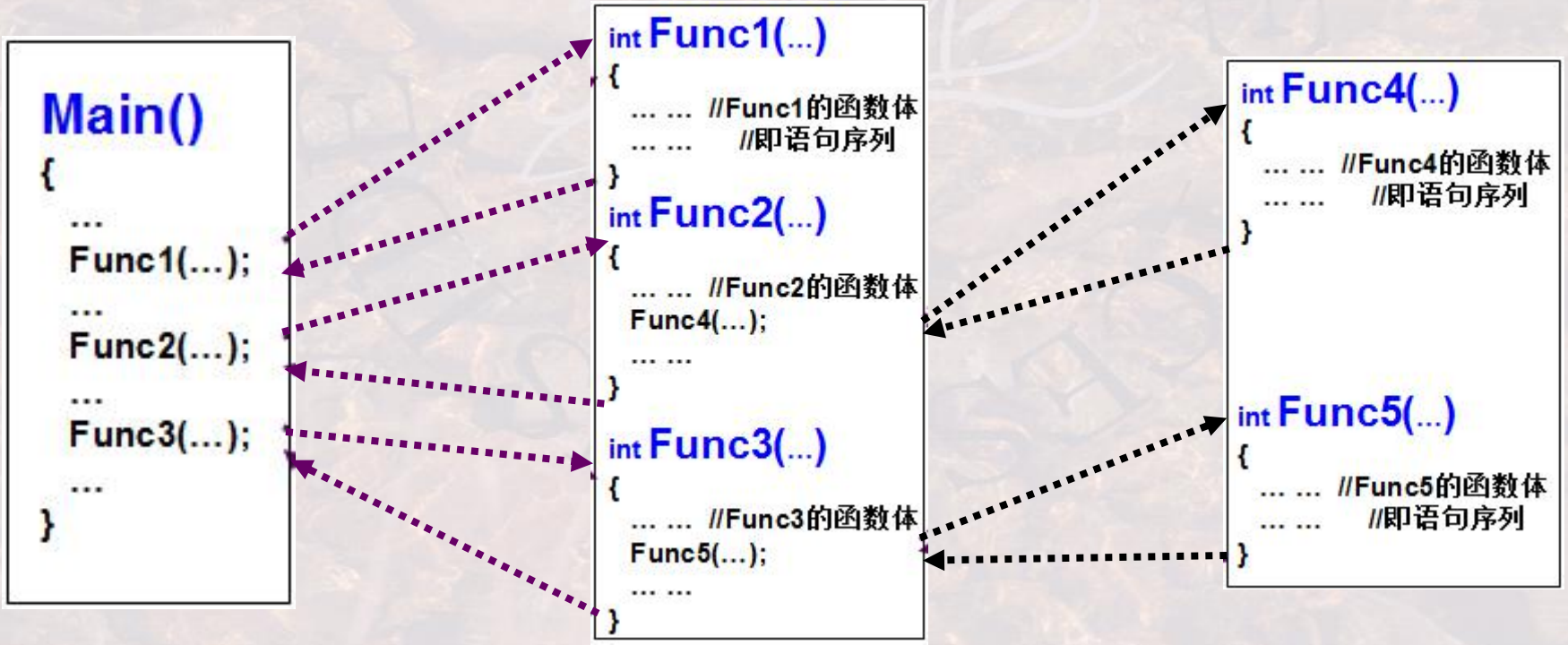
3.1.5. 程序构造及其表达方法

（1） 程序是构造的，不是编的

传统程序构造及其表达方法----由粗到细

为控制复杂性，先以函数来代替琐碎的细节，着重考虑函数之间的关系，以及如何解决问题

在前一阶段考虑清楚后或编制完成后，再编写其中的每一个函数。而函数的处理同样采取这种思路



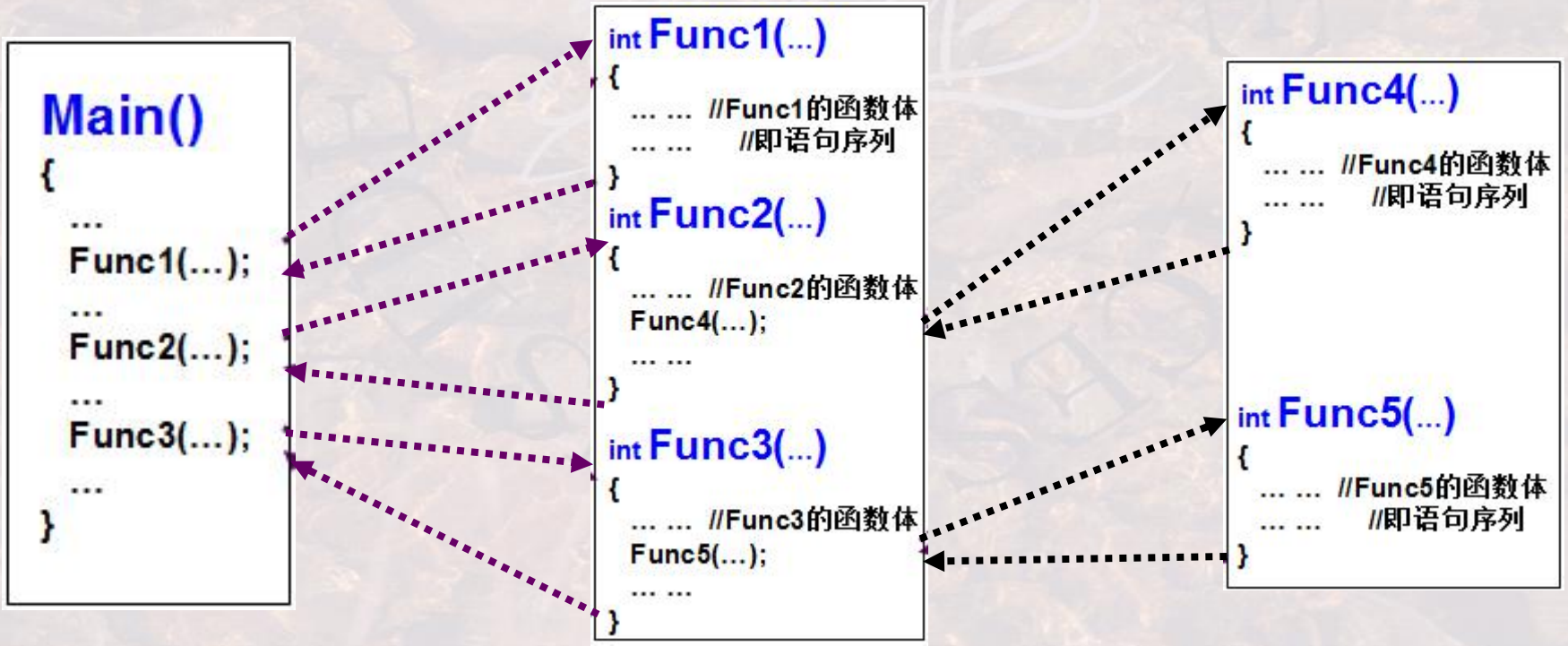
3.1.5. 程序构造及其表达方法

（1） 程序是构造的，不是编的

传统程序构造及其表达方法----也可以由细到粗

上一层次的函数依据下层函数来编写，确认正确后再转至更上层问题处理

首先编写一些基础性的函数，并确定其正确后，再处理上一层次的问题。



算法(求解问题的一系列步骤)与程序的基本控制结构

✓顺序结构：“执行A，然后执行B”，是按顺序执行一条条规则的一种结构 。

✓分支结构：“如果Q成立，那么执行A，否则执行B”，Q是某些逻辑条件，即按条件判断结果决定执行哪些规则的一种结构 。

✓循环结构：控制指令或规则的多次执行的一种结构---迭代(iteration)

◆循环结构又分为有界循环结构和条件循环结构。

✓有界循环：“执行A指令N次”，其中N是一个整数。

✓条件循环：某些时候称为无界循环，“重复执行A直到条件Q成立”或“当Q成立时反复执行A”，其中Q是条件。

算法与程序构造的表达方法：程序流程图

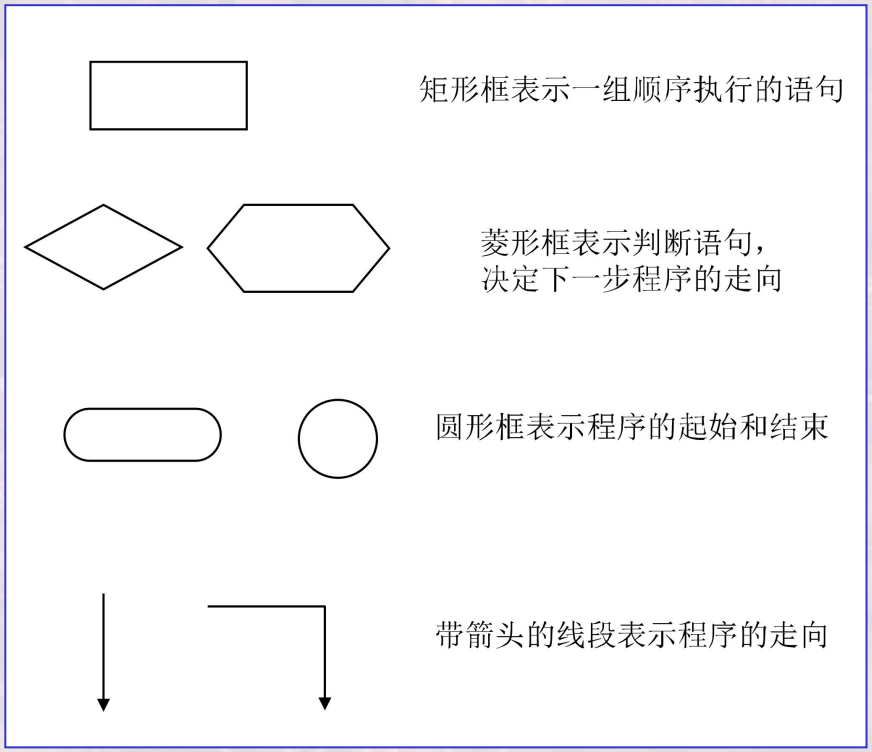
流程图的基本表示符号

✓**矩形框**：表示一组顺序执行的规则或者程序语句。

✓**菱形框**：表示条件判断，并根据判断结果执行不同的分支。

✓**圆形框**：表示算法或程序的开始或结束。

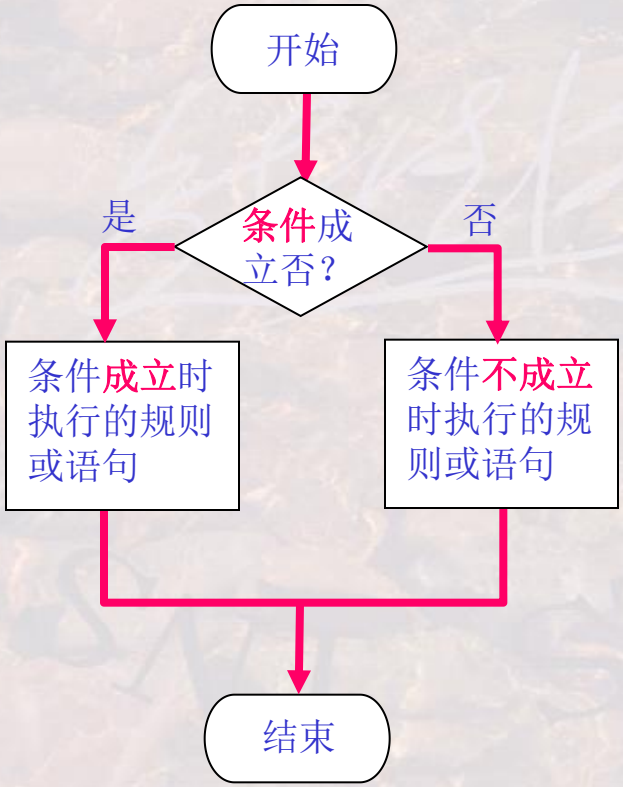
✓**箭头线**：表示算法或程序的走向。



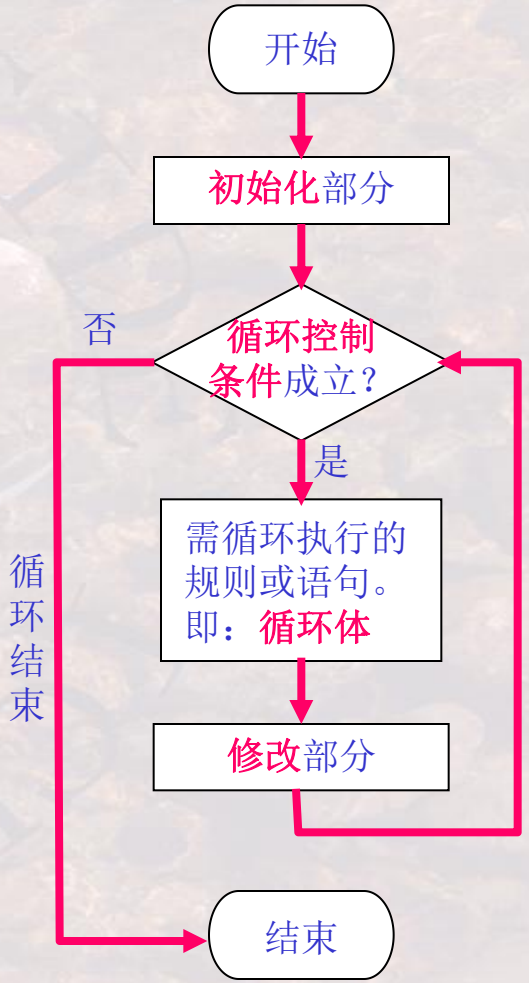
算法与程序构造的表达方法：程序流程图

◆三种控制结构的流程图表示方法示意

顺序结构的流程图



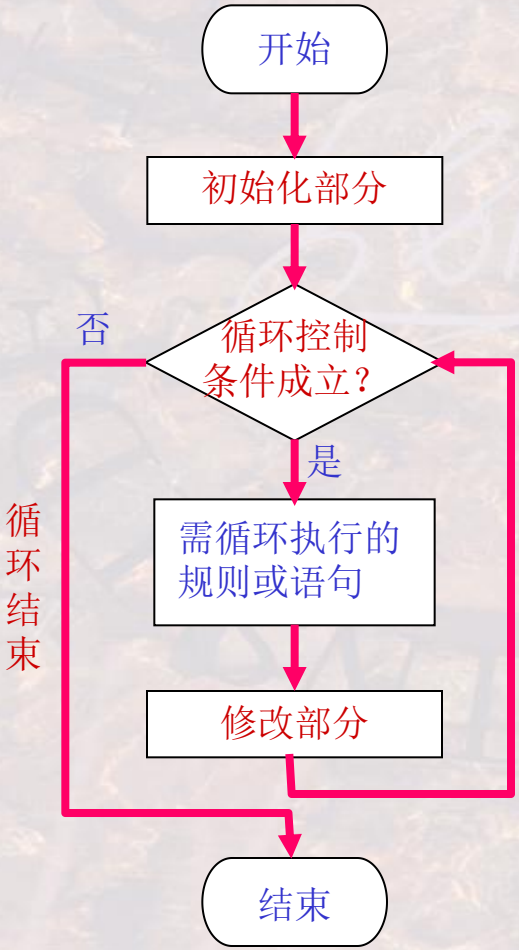
分支结构的流程图



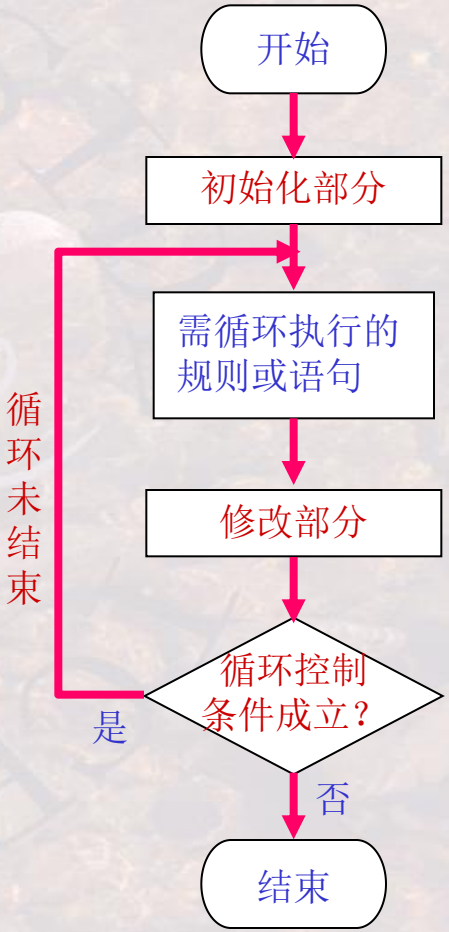
循环结构的流程图

算法与程序构造的表达方法：程序流程图

◆循环结构的两种情况的流程图表示方法示意



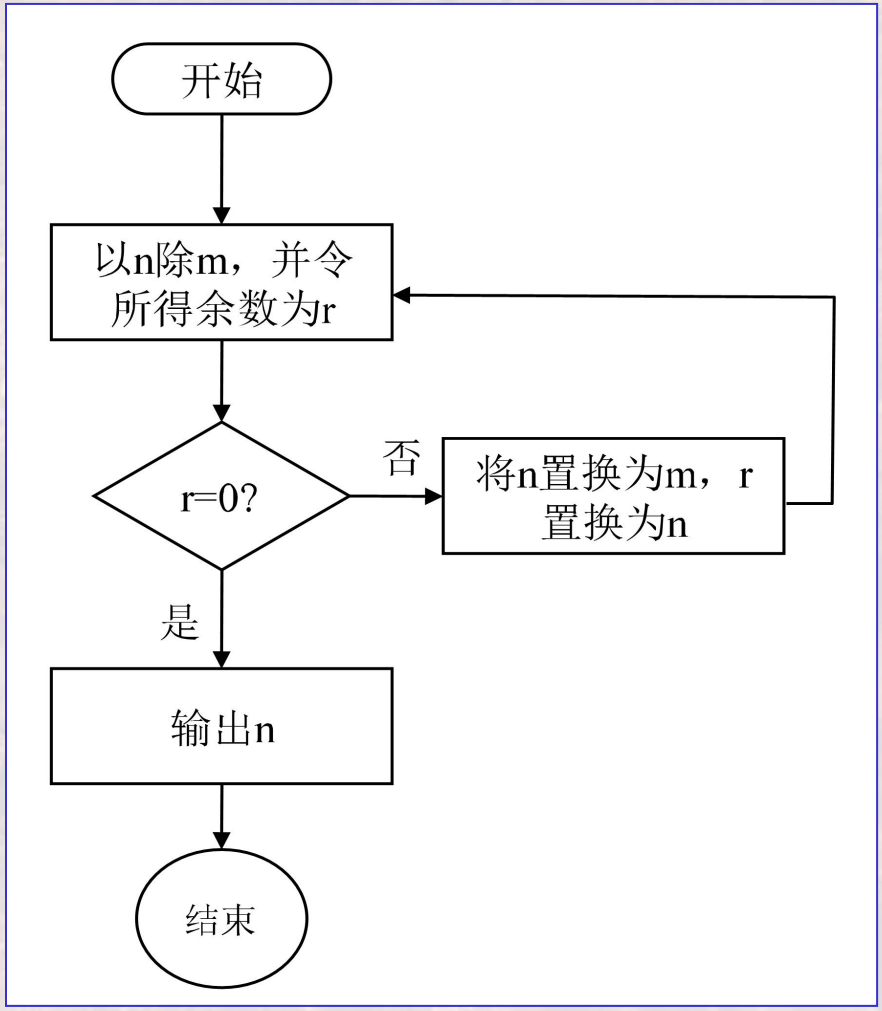
有界循环结构的流程图(循环次数确定)



条件循环结构的流程图(循环次数不确定)

算法与程序构造的表达方法：程序流程图

◆欧几里德算法流程图



算法与程序构造的表达方法：步骤描述法

◆步骤描述法，即用人们日常使用的语言和数学语言描述算法的步骤。

◆例如：sum=1+2+3+4+…+n求和问题的算法描述

Start of the algorithm(算法开始)

(1)输入N的值；

(2)设 i 的值为1；sum的值为0；

(3)如果 i<=N，则执行第(4)步，否则转到第(7)步执行；

(4)计算 sum + i，并将结果赋给sum；

(5)计算 i+1，并将结果赋给i;

(6)返回到第3步继续执行；

(7)输出sum的结果。

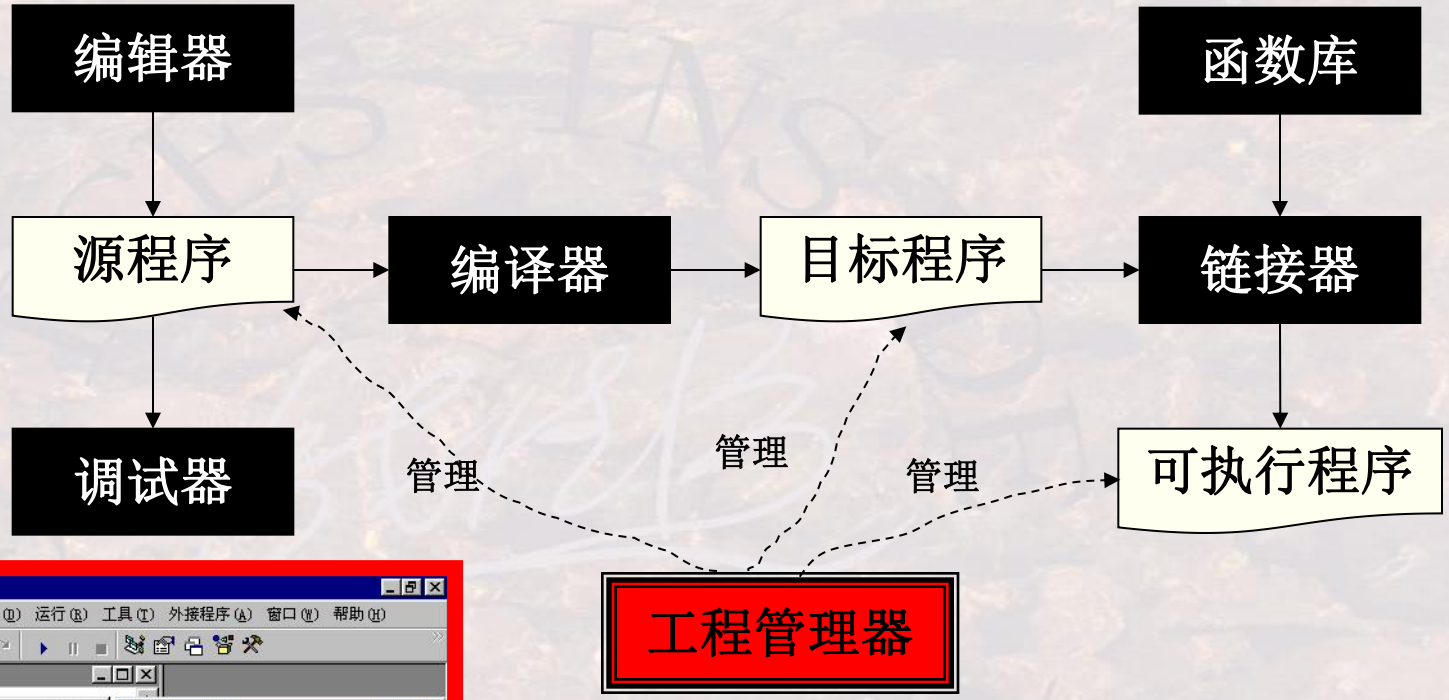
End of the algorithm(算法结束)

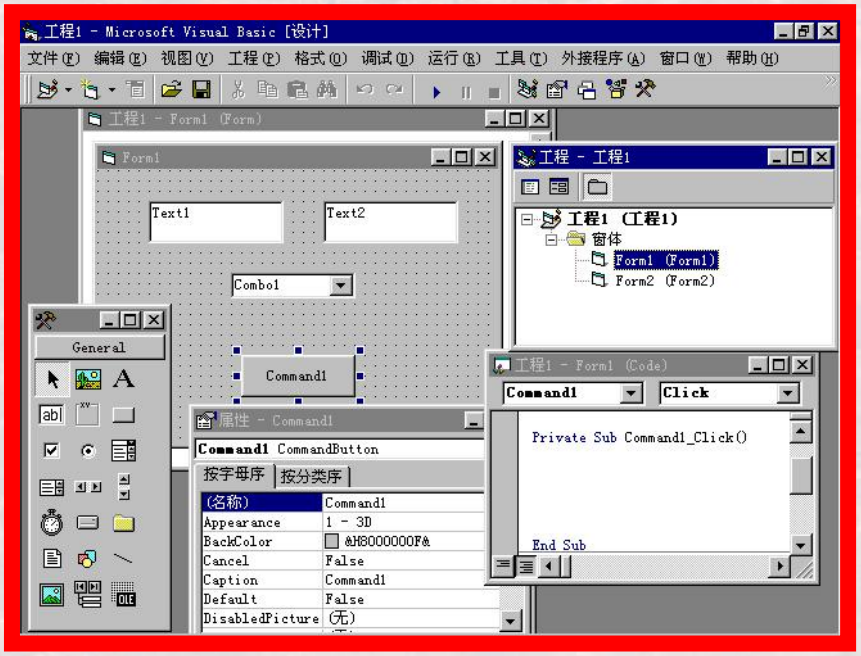
自然语言表示的算法容易出现二义性、不确定性等问题

3.1.5. 程序构造及其表达方法

（7）程序开发环境包括哪些部分呢？

程序开发环境





3.3 现代程序的基本构成要素

类与对象的概念

- 现实世界(或者说系统)是由可区分“对象”构成的...(对象标识)
- “对象”之间是相互独立的，对象有自己的状态、对象自己执行自身的操作而无需其他对象干预...(对象属性及函数)
- “对象”在接收到需服务的请求(消息/事件)时，可为其他对象提供服务...(消息驱动/事件驱动)

类与对象的概念

●有些对象具有相似的特性描述，组成一个“类”。

●类是对象的抽象或称对象的“类型”，定义了同类型对象的框架(名字、属性和功能);

●对象是类的实例，是实际运行的个体。

类

类名: <>
(类的属性)
性别: <>
身高: <>
体重: <>
(类的功能)
回答身高(): <>
回答体重(): <>
回答性别(): <>

同类对象的共性形式或者说对象的类型：类

各自独立运行的类的实例：对象

对象

对象名: 张三
(对象的属性)
性别: 男
身高: 1.80m
体重: 70kg
(对象的功能)
回答身高(): 身高是1.80m
回答体重(): 体重是70kg
回答性别(): 性别是男

张三

.....

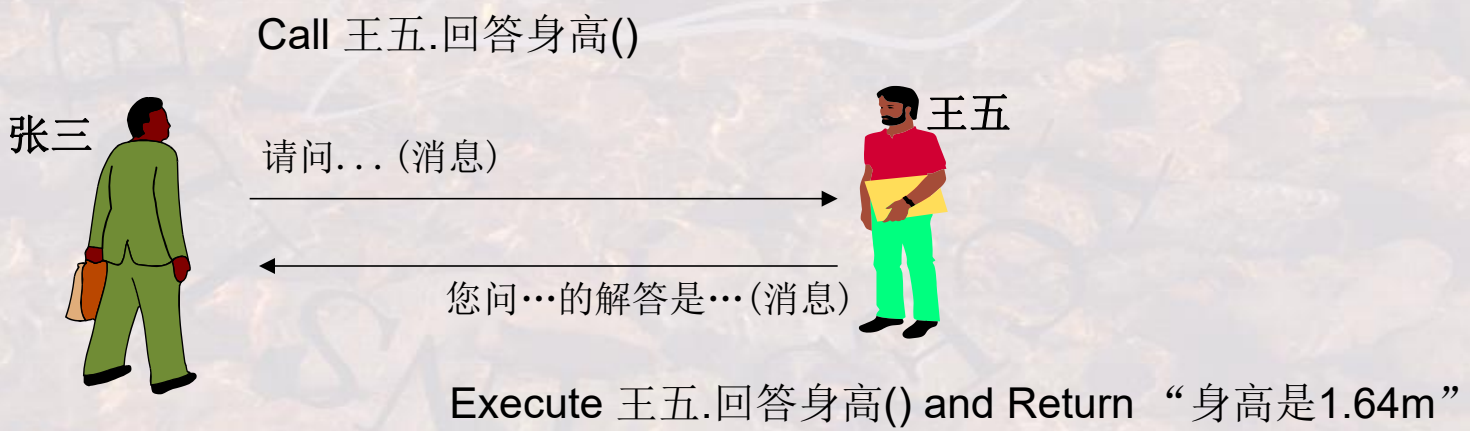
对象名: 王五
(对象的属性)
性别: 男
身高: 1.64m
体重: 62kg
(对象的功能)
回答身高(): 身高是1.64m
回答体重(): 体重是62kg
回答性别(): 性别是男

王五

3.3.1 对象与类的概念---通俗示例相关概念
(2)对象间的交互--消息

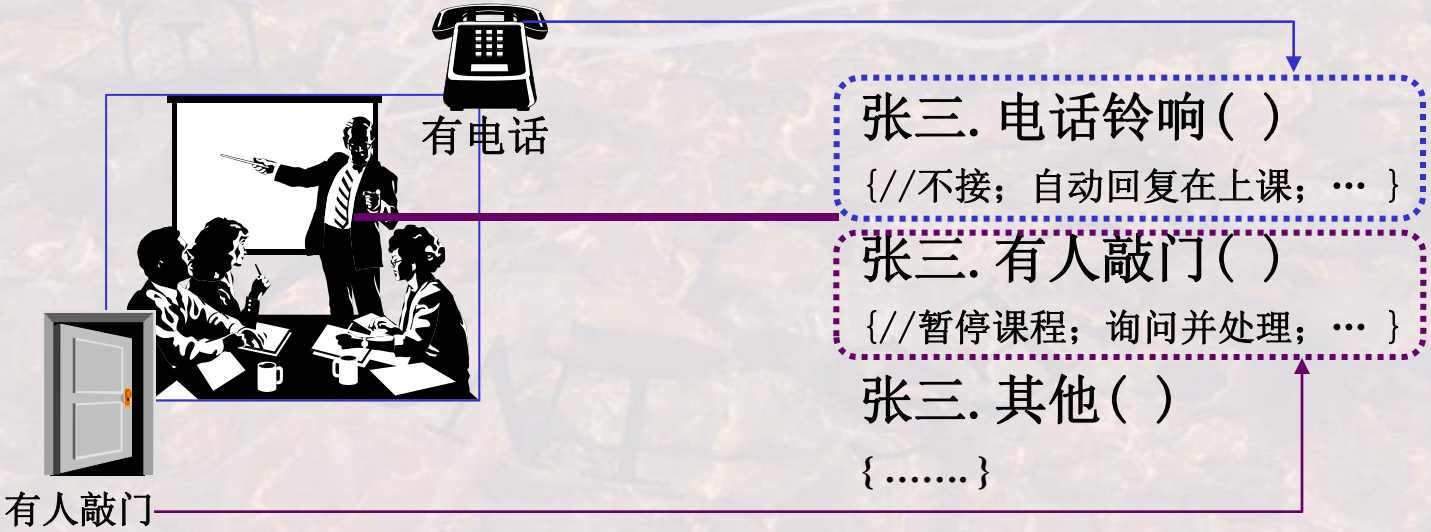
消息

- **消息**是对象之间传递的内容，如指示、打听、请求 … …
- **消息是对象之间交互的唯一的内容**
- 消息是 **“对象.函数()”**的调用和执行



事件(Event)

●事件是外界产生的一种能够激活对象功能的特殊消息。当发生事件后将给所涉及对象发送一个消息，对象便可执行相应的功能---事件驱动；



3.3.2 类与对象的概念--面向对象的程序

(1)面向过程的计算机语言 vs. 面向对象的计算机语言

(传统的、面向过程的)计算机语言(程序)的基本构成要素

程序：变量与常量、表达式、语句与函数。

```
main()
{
  int D1，  D2;
  …
  f1();
  …
}

int f1()  //独立的函数
{
  …  //f1()的程序语句集合
}
```

主函数main()：主程序

函数调用：调用一个函数完成一个计算

函数定义：定义一个新函数，即自己编写的程序段落

传统程序

(现代的、面向对象的)计算机语言(程序)的基本构成要素

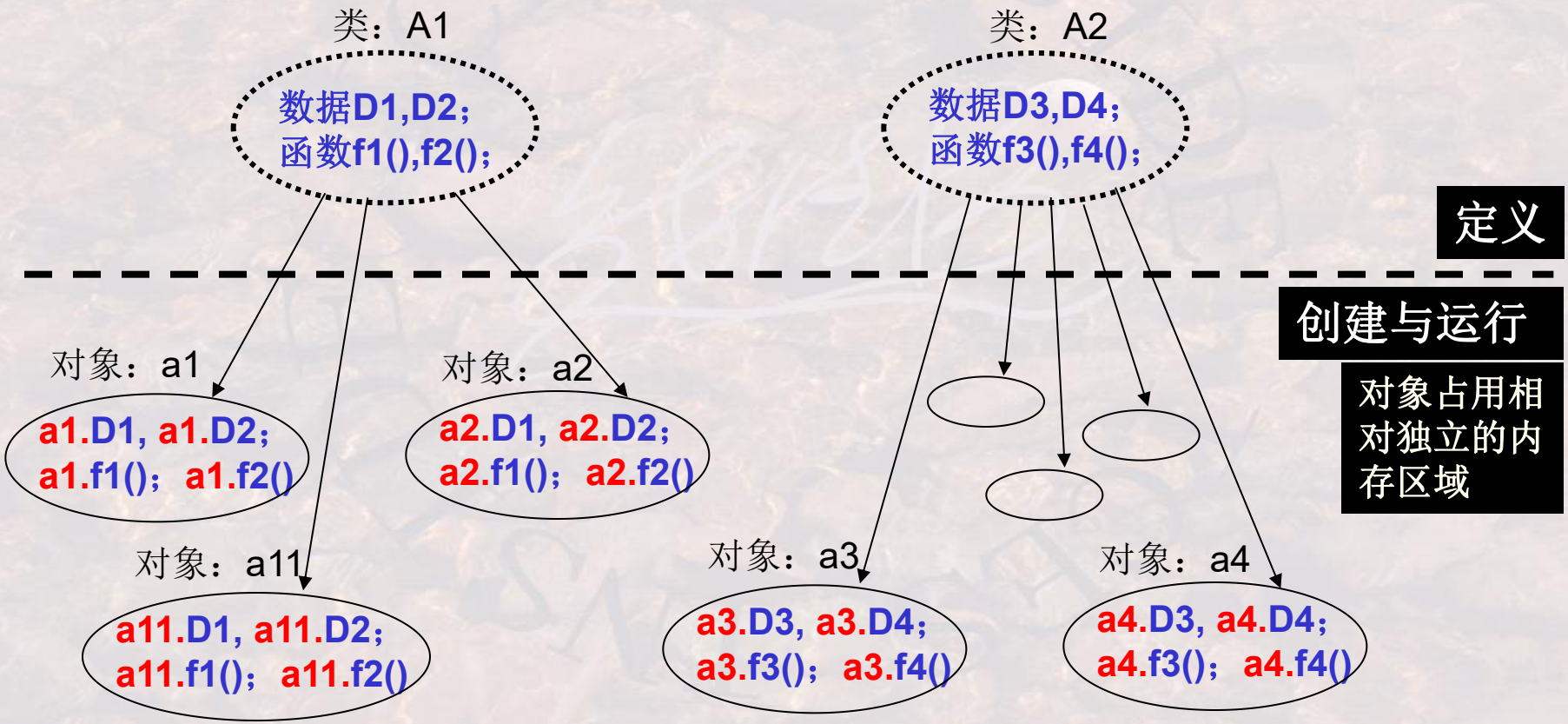
类与对象，消息与事件，函数或称方法

3.3.2 类与对象的概念--面向对象的程序

(2)对象和类要解决什么问题？

“若干同类别对象，虽有相同的程序，但却处理不同的数据而产生不同的结果呢？”

两方面技术：一是封装，二是对象的自动产生与运行。

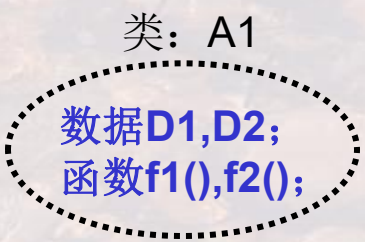


数据结构封装了若干个变量；函数封装了若干行的语句；

类/对象封装了若干数据结构和函数。

3.3.2 类与对象的概念--面向对象的程序
(3)怎样定义对象的结构---定义“类”？

对象结构的定义---即类的定义

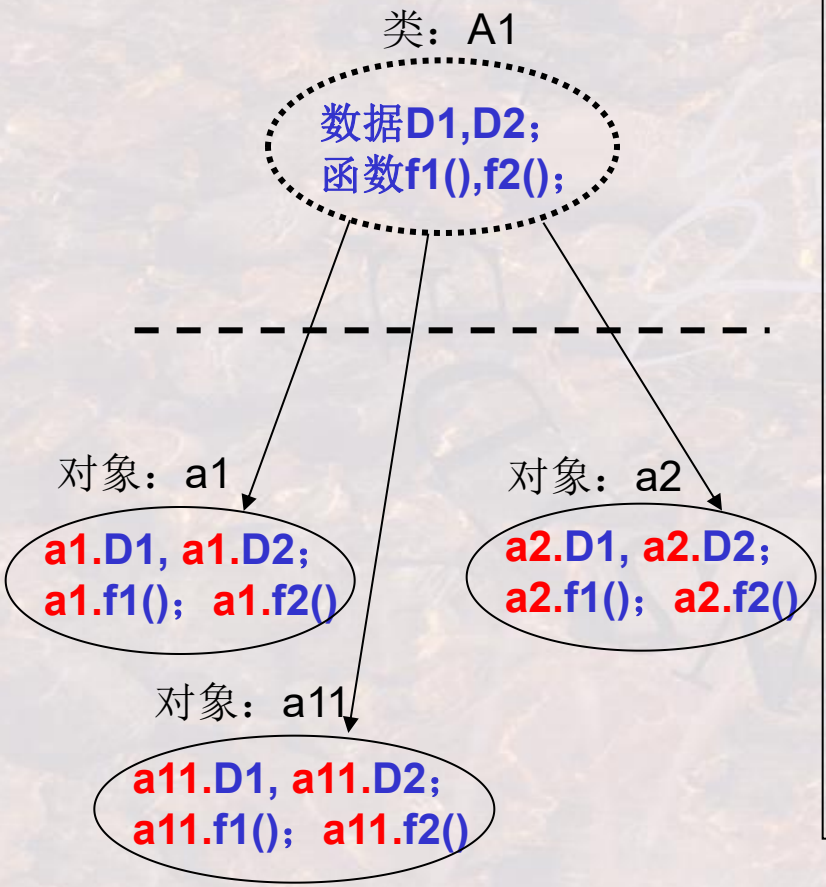



```
Class  A1                  //定义一个类
{
    int  D1, D2             //定义变量
    int  f1();              //定义类中的函数
    int  f2();              //定义类中的函数

    int  f1()               //类A1的函数f1的程序
    {  //f1的程序语句块 }

    int f2()                //类A1的函数f2的程序
    {   //f2的程序语句块1
        a4 = new  A2;     //创建类A2的对象a4
       //f2的程序语句块2
       D1 = call  a4.f3(); //调用另一对象的某一函数
       //f2的程序语句块3 }
}
```

对象的创建与运行




```
Main()
{  int M1, M2;
   a1 = new A1;        //创建类A1的对象a1
   a2 = new  A1;       //创建类A1的对象a2
   a11 = new A1;       //创建类A1的对象a11

   M1 = call a1.f1();          //执行对象a1的f1的程序
   M2 = call a2.f1();          //执行对象a2的f1的程序
   a1.D1 = 15;         //对象a1的数据D1被赋值为15
   a2.D1 = 20;         //对象a2的数据D1被赋值为20
   … …
}
```

3.3.2 类与对象的概念--面向对象的程序

(5)再看：面向过程的计算机语言 vs. 面向对象的计算机语言

差别：面向过程的程序要素 vs. 面向对象的程序要素

```
Main()
{
    int  D1, D2
    int sum; sum=D1;
    call f1(sum);
    call f2();
}

int f1(int qty)
{  int temp;
   temp=qty*qty;
   return(qty);
    //f1的程序语句块 }

int f2()
{ //f2的程序语句块 }
```

全局变量

独立定义的函数

局部变量

```
Class  A1
{     int  D1, D2;
      int  f1(int qty)
      {   int temp; … …//f1的程序语句块  }
}
int A1:: f2()
{  //f2的程序语句块 }
int f2() { //f2的程序语句块 }
Main() {  …
    a1 = new A1;
    a2 = new A1;
    a1.D1 = M1;
    call f2();
    call  a1.f1(M1);
    call  a2.f1(M2);
}
```

定义“类”

类中定义的函数

类中定义的函数

独立定义的函数

用类产生对象

调用独立的函数

调用对象的函数

消息与对象的交互:

消息即“传送的数据”，“函数调用”。 方法是响应消息的函数体