

第7讲 程序编写与计算机语言

李玉华

华中科技大学计算机学院智能与分布计算实验室

ldcliyuhua@hust.edu.cn,
<http://idc.hust.edu.cn/~yhli/>

第7讲 程序编写与计算机语言

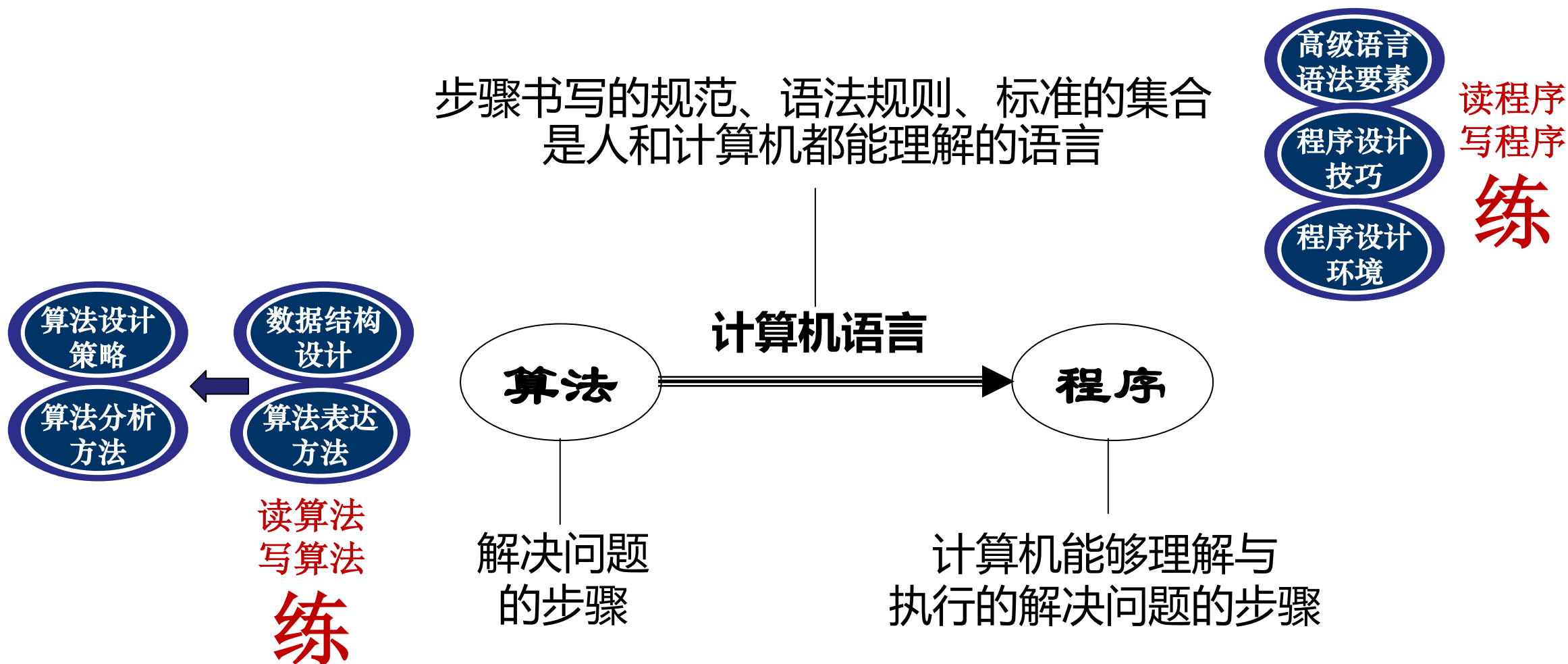
2

- 一、计算机语言的发展**
- 二、高级语言（程序）的基本构成要素
- 三、不同的计算机语言之异同点
- 四、用高级语言构造程序
- 五、计算机语言与编译器

计算机语言的发展

3

(1) 算法、计算机语言与计算机程序



计算机语言的发展

4

(2) 机器语言与机器程序

指令
系统

CPU用二进制和编码提供的可以解释并执行的命令的集合

操作码	地址码
100001	10 00000111
100010	11 00001010

机器
语言

用二进制和编码方式提供的指令系统所编写程序的语言

计算7+10并存储的机器程序

100001	10
00000111	
100010	10
00001010	
100101	11
00000110	
111101	00

所有程序都需转换成机器程序，计算机才能执行

计算机语言的发展

5

(3) 汇编语言

怎样解决机器语言编写程序所存在的困难--符号化语言?

◆用符号编写程序 ==> 翻译 ==> 机器语言程序

◆人们提供了用助记符编写程序的规范/标准。同时开发了一个翻译程序（被称为汇编程序），实现了将符号程序自动转换成机器语言程序的功能

操作码	地址码
100001	1000000111

↓

MOV A, 7

计算7+10并存储的汇编语言源程序

```
MOV A, 7
ADD A, 10
MOV (6), A
HLT
```



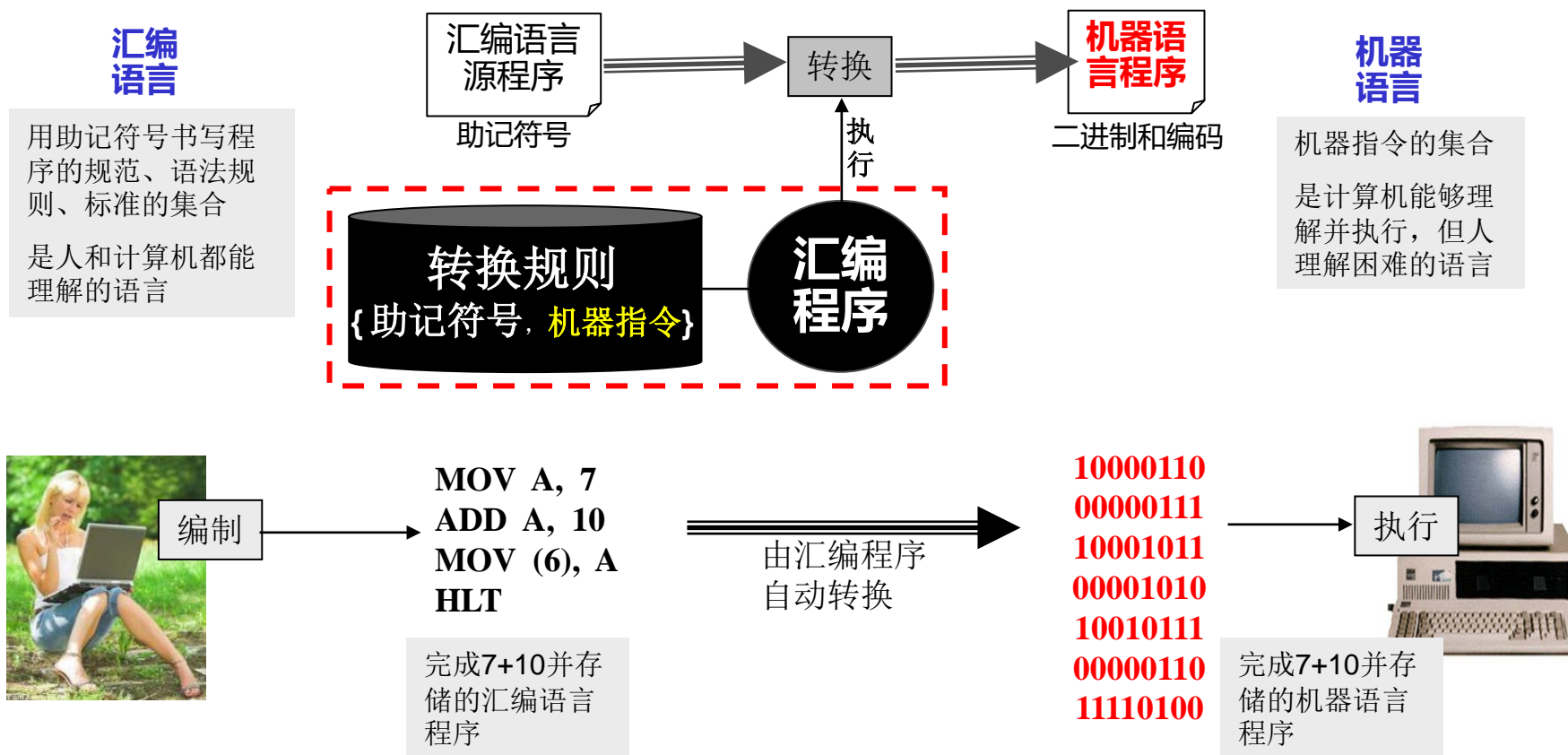
用助记符编写程序的语言，可与机器语言一一对应

计算机语言的发展

6

(3) 汇编语言

符号化程序机器不能直接执行怎么办--汇编/翻译



计算机语言的发展

7

(4) 高级语言

程序编写能否更便捷？

◆人们提供了类似于自然语言方式、以语句为单位书写程序的规范/标准。并开发了一个翻译程序，实现了将语句程序自动翻译成机器语言程序的功能。



用类似自然语言的语句编写程序的语言。

计算7+10并存储的高级语言(源)程序

```
Result = 7+10;  
Return
```

高级语言源程序：是用高级语言编出的程序。

编译程序：是将高级语言源程序翻译成机器语言程序的程序。

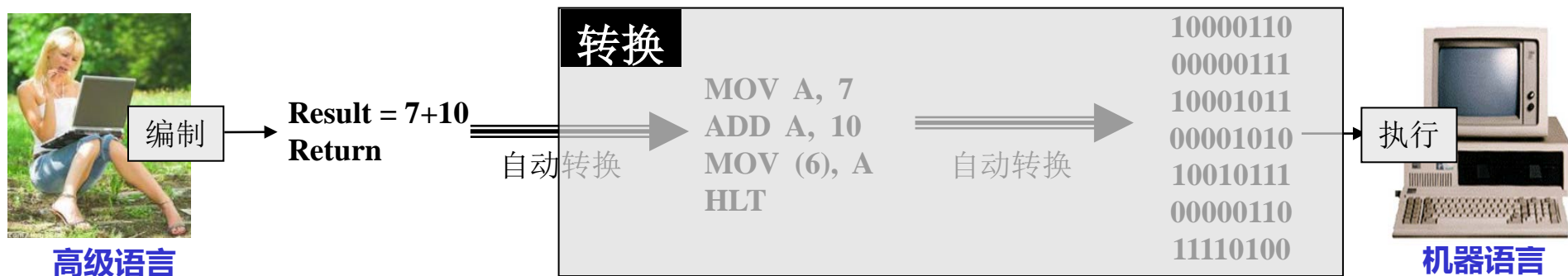
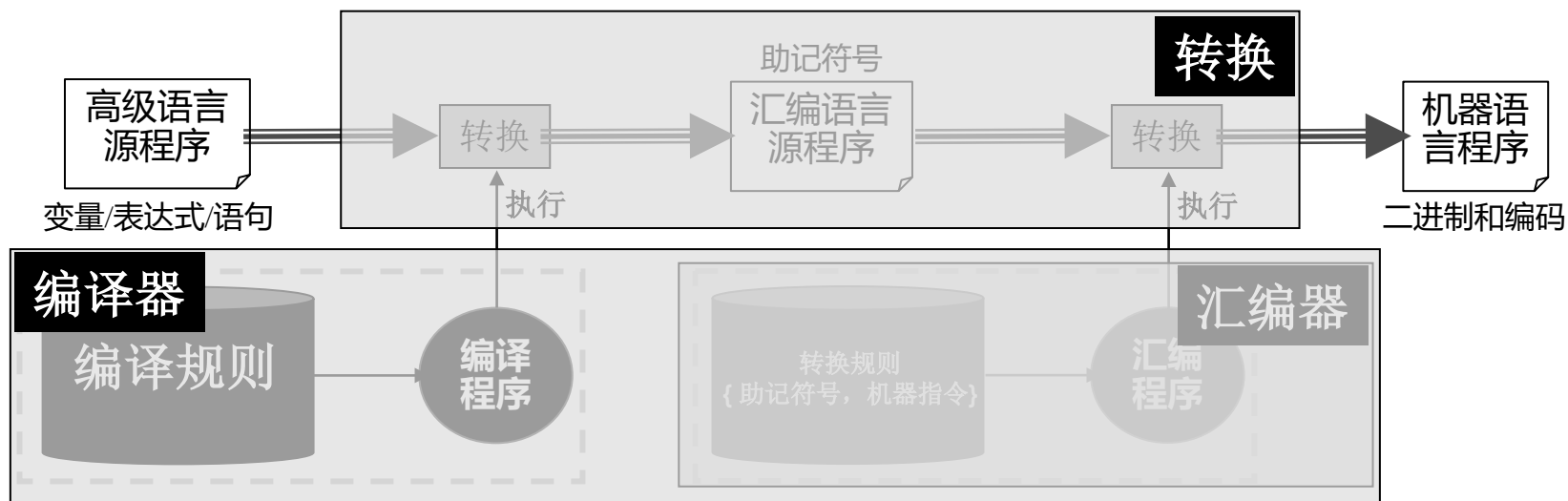
计算机语言的发展

8

(5) 高级语言 vs. 汇编语言

高级语言：机器无关性；一条高级语言语句往往可由若干条机器语言语句实现且不具有对应性

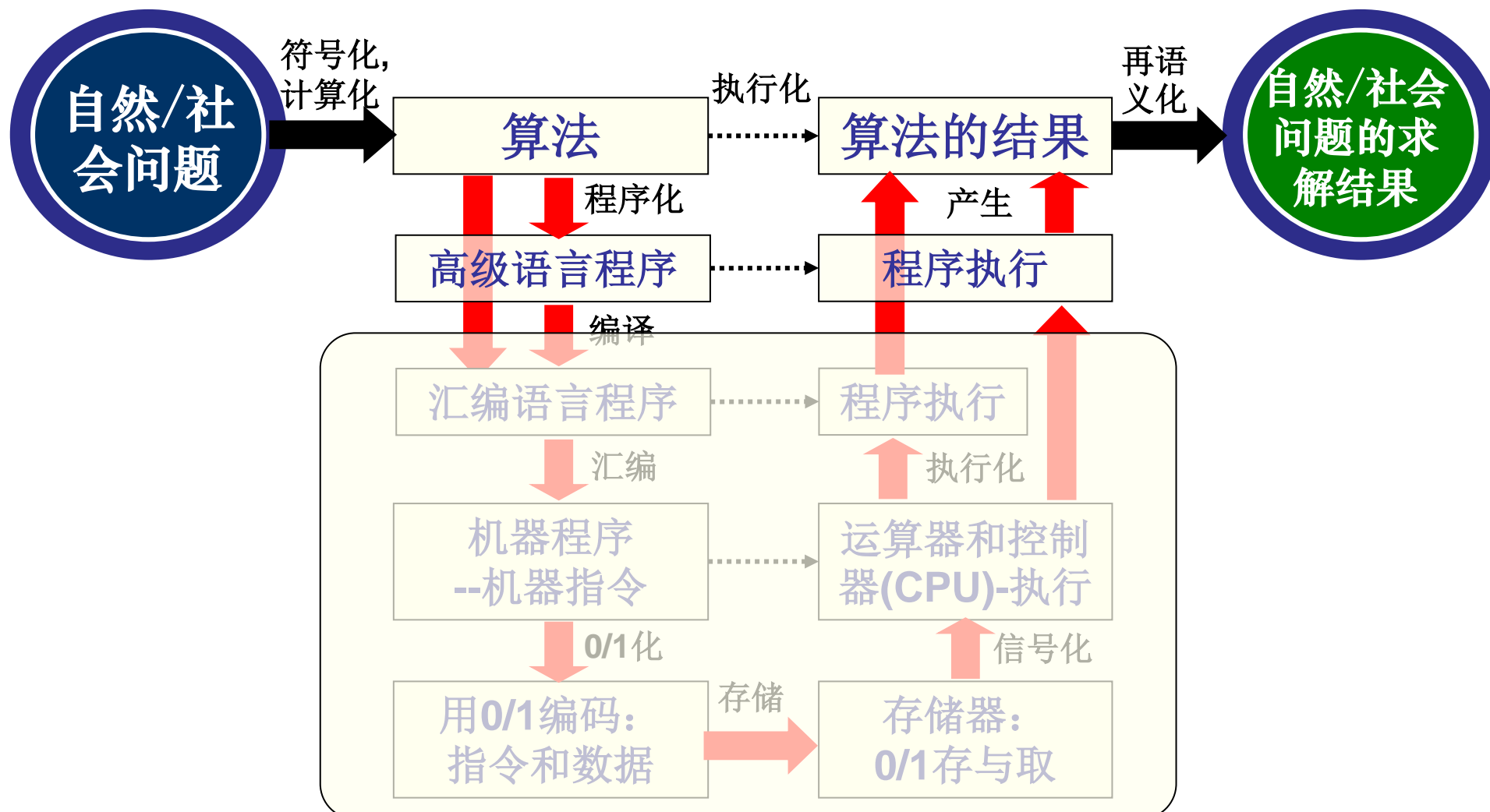
汇编语言：机器相关性；汇编语言语句和机器语言语句有对应性



计算机语言的发展

9

(6) 用高级语言进行问题求解的实现过程



计算机语言的发展

10

(7) 高级语言 (源) 程序

符号化

语句化

结构化

```
K = 0;
```

```
For I = 1 to 100 Step 1
```

```
{ If I <= 50 && I > 30
```

```
  { K = K + I; }
```

```
}
```

编译化

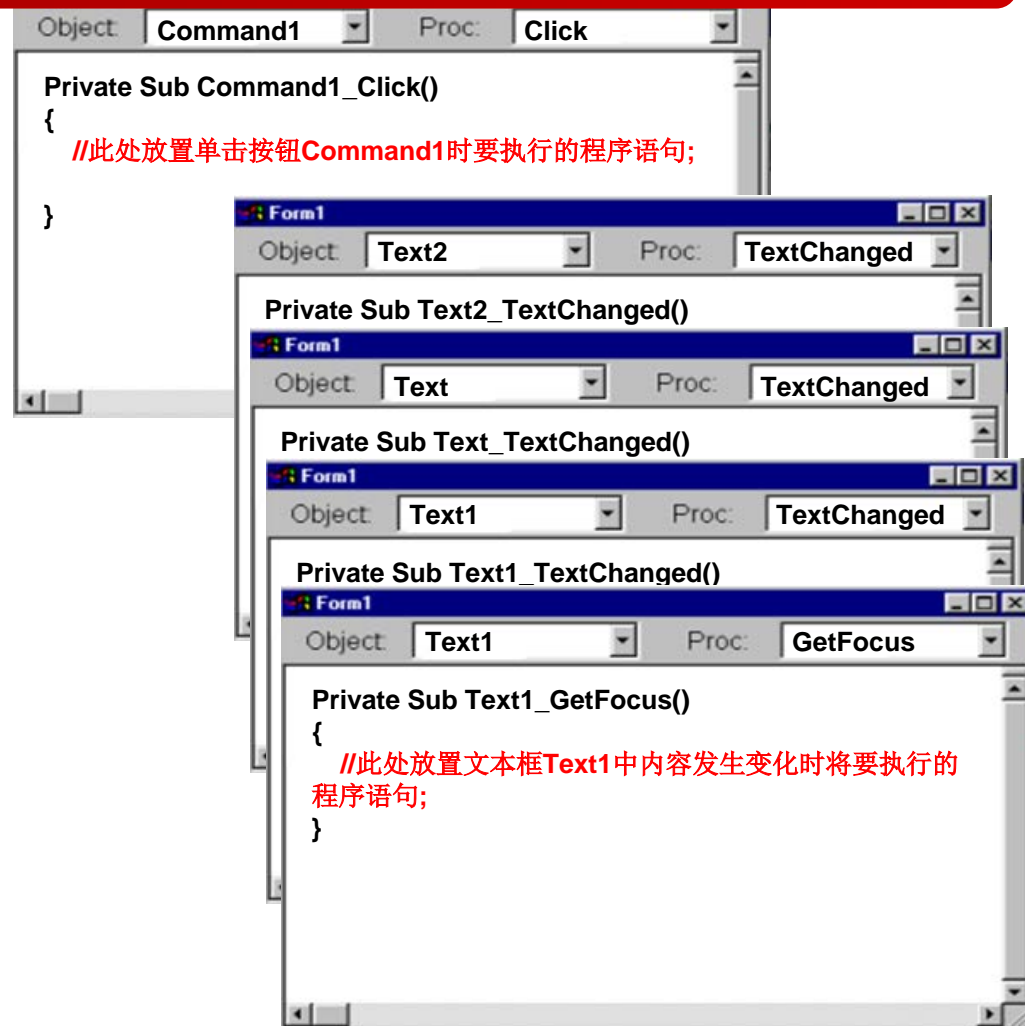
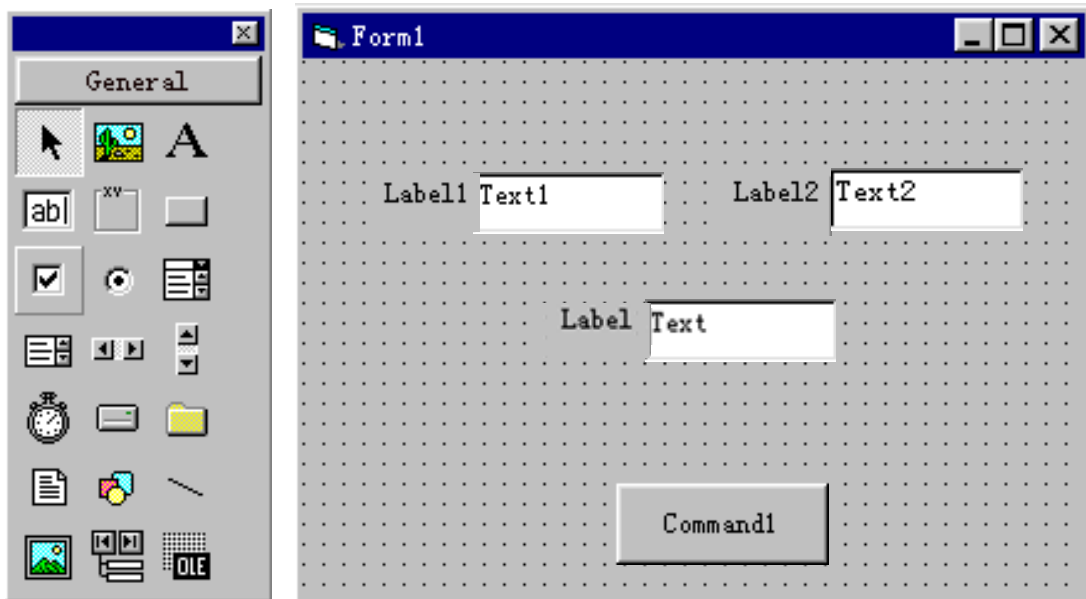
机器语言程序

计算机语言的发展

11

(8) 面向对象的程序设计语言与可视化构造语言

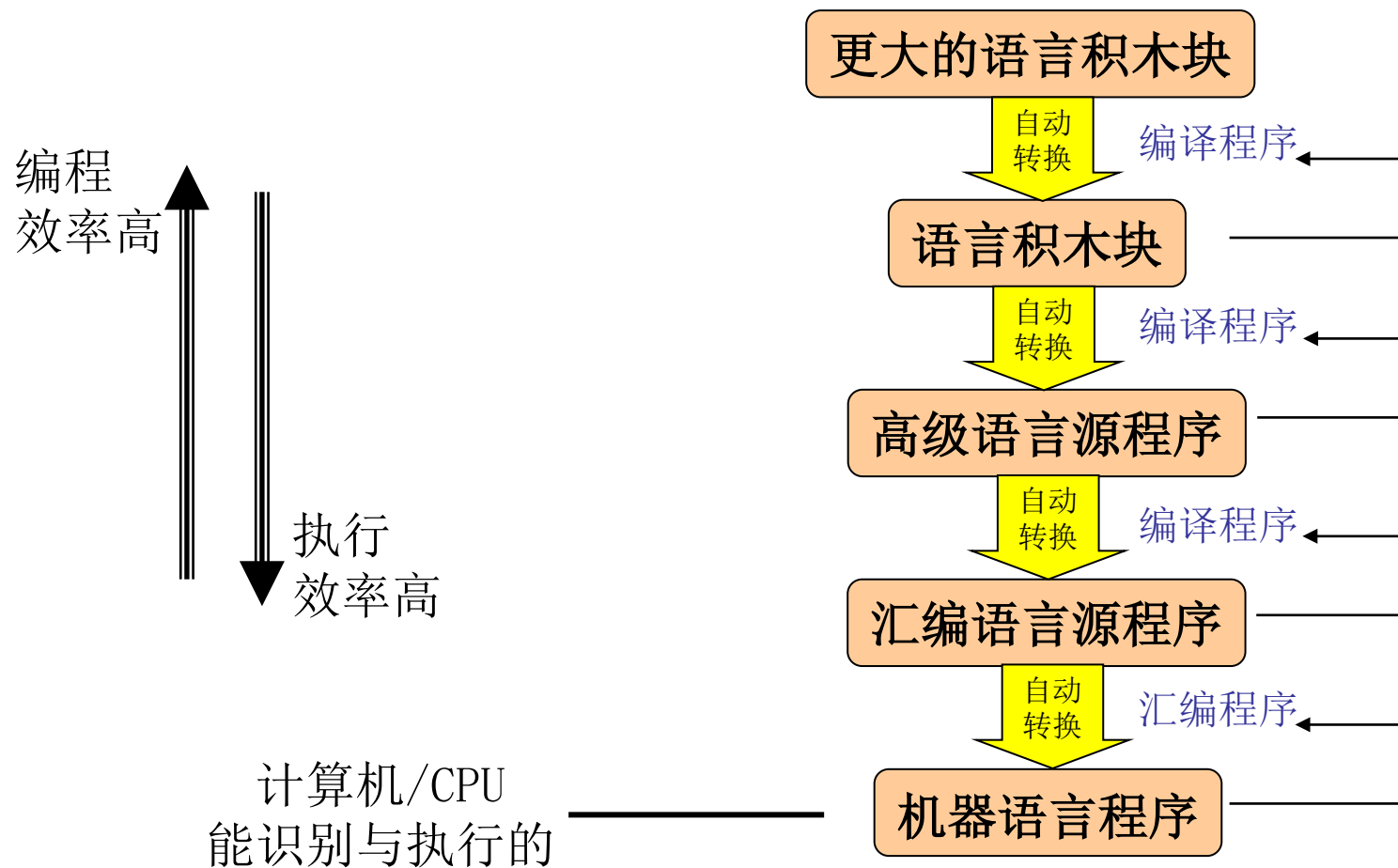
----像堆积木一样构造程序



计算机语言的发展

12

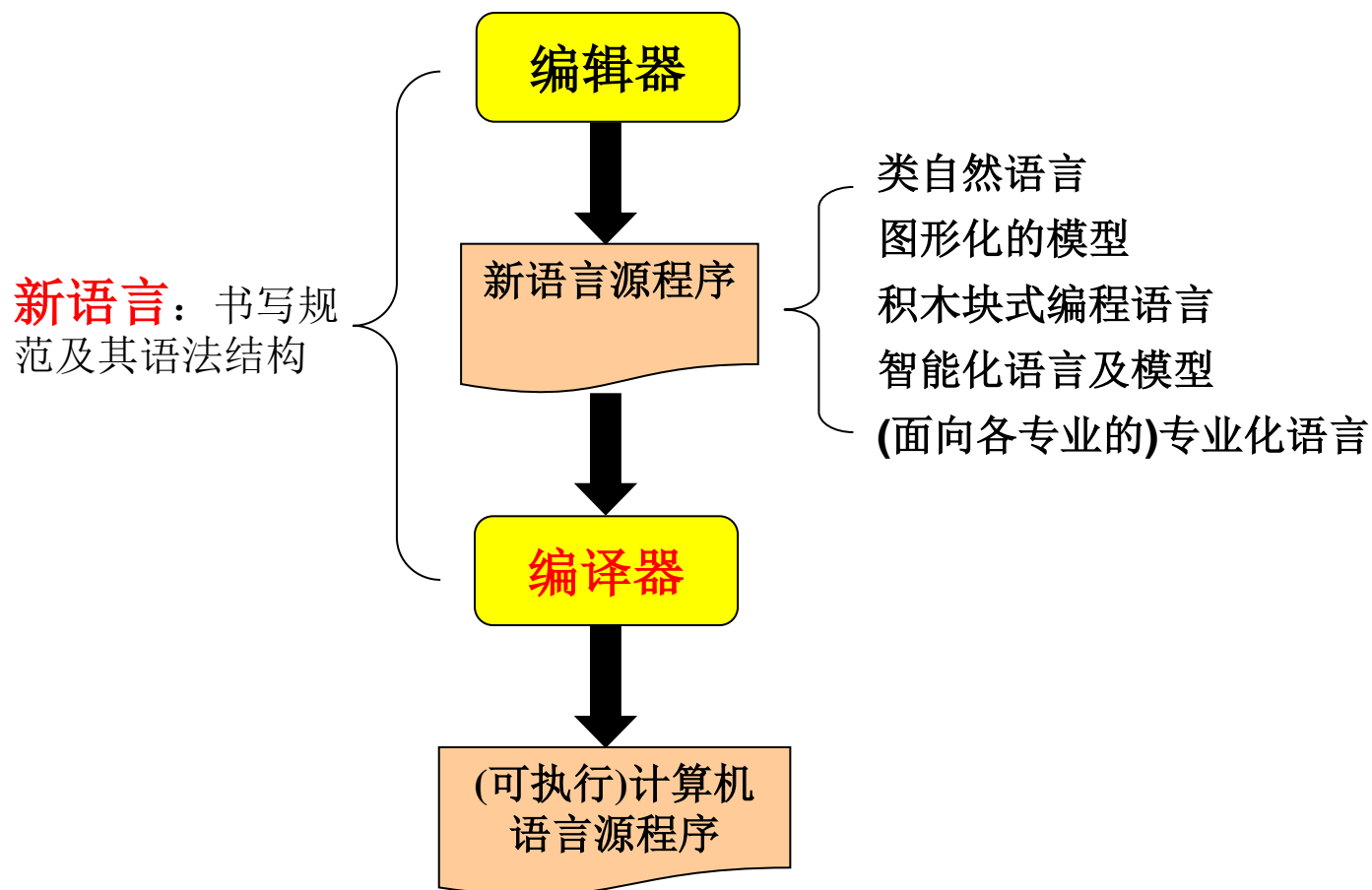
(9) 计算机语言发展的基本思维



计算机语言的发展

13

(10) 不仅要用语言，还要发明新语言



计算机语言的发展

14

(11) 计算机技术是伴随着计算机语言的不断发展而发展起来的

因计算机语言
获得图灵奖的

- 1966 A.J. Perlis: 编程技术和编译架构
- 1972 E.W. Dijkstra: ALGOL语言
- 1974 Donald E. Knuth: 程序语言
- 1977 John Backus : 高级语言, Fortran
- 1979 Kenneth E. Iverson: 编程语言, APL
- 1980 C. Antony R. Hoare: 编程语言
- 1981 Edgar F. Codd: 关系数据库语言
- 1984 Niklaus Wirth: 开发了EULER、ALGOL-W、MODULA和PASCAL一系列崭新的计算语言。
- 1987 John Cocke: 编译器
- 2001 Ole-Johan Dahl、Kristen Nygaard: 面向对象编程, SIMULA I 和SIMULA 67中。
- 2003 Alan Kay :面向对象语言, Smalltalk
- 2005 Peter Naur: Algol60程序语言。
- 2006 Fran Allen: 编译器

计算机语言的发展

15

(12) 多种多样的高级语言

Mother Tongues

Tracing the roots of computer languages through the ages

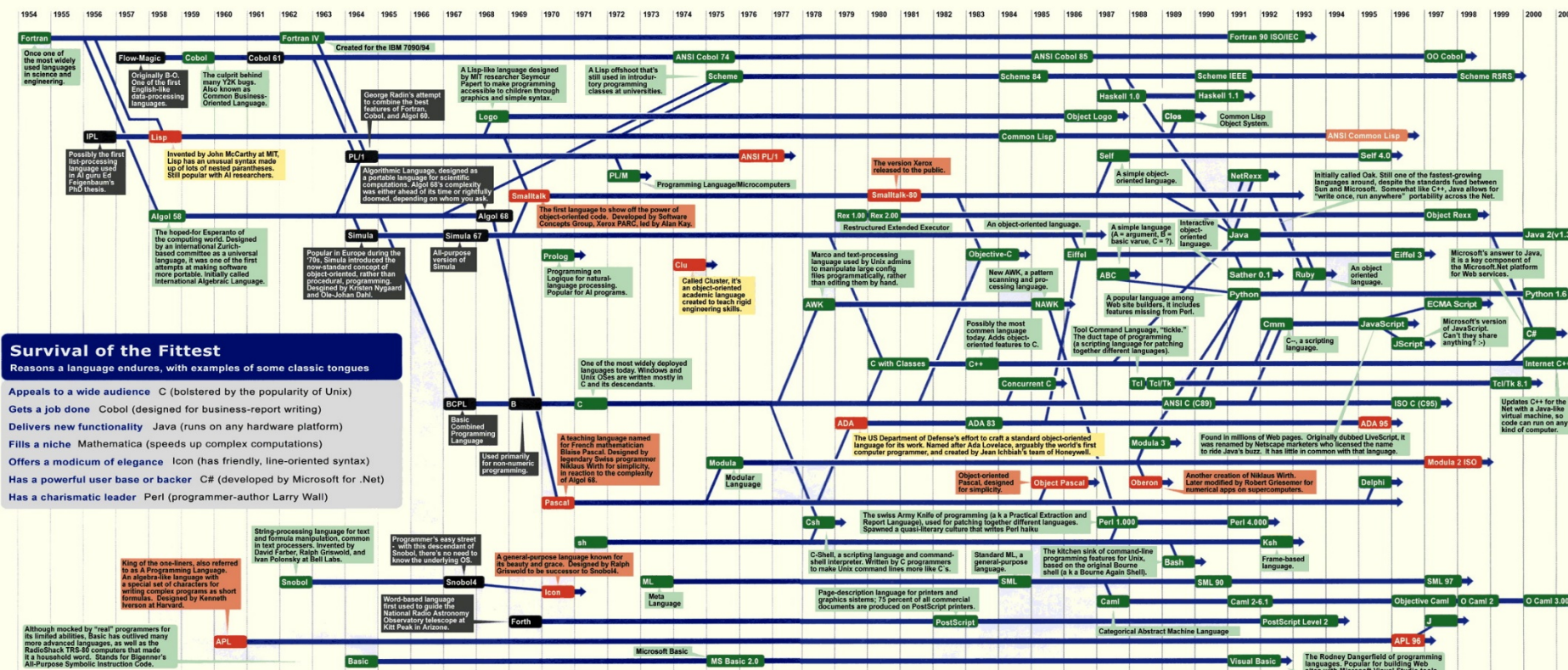
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at [HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html](http://www.informatik.uni-freiburg.de/Java/misc/lang_list.html). - Michael Mendeno

Key

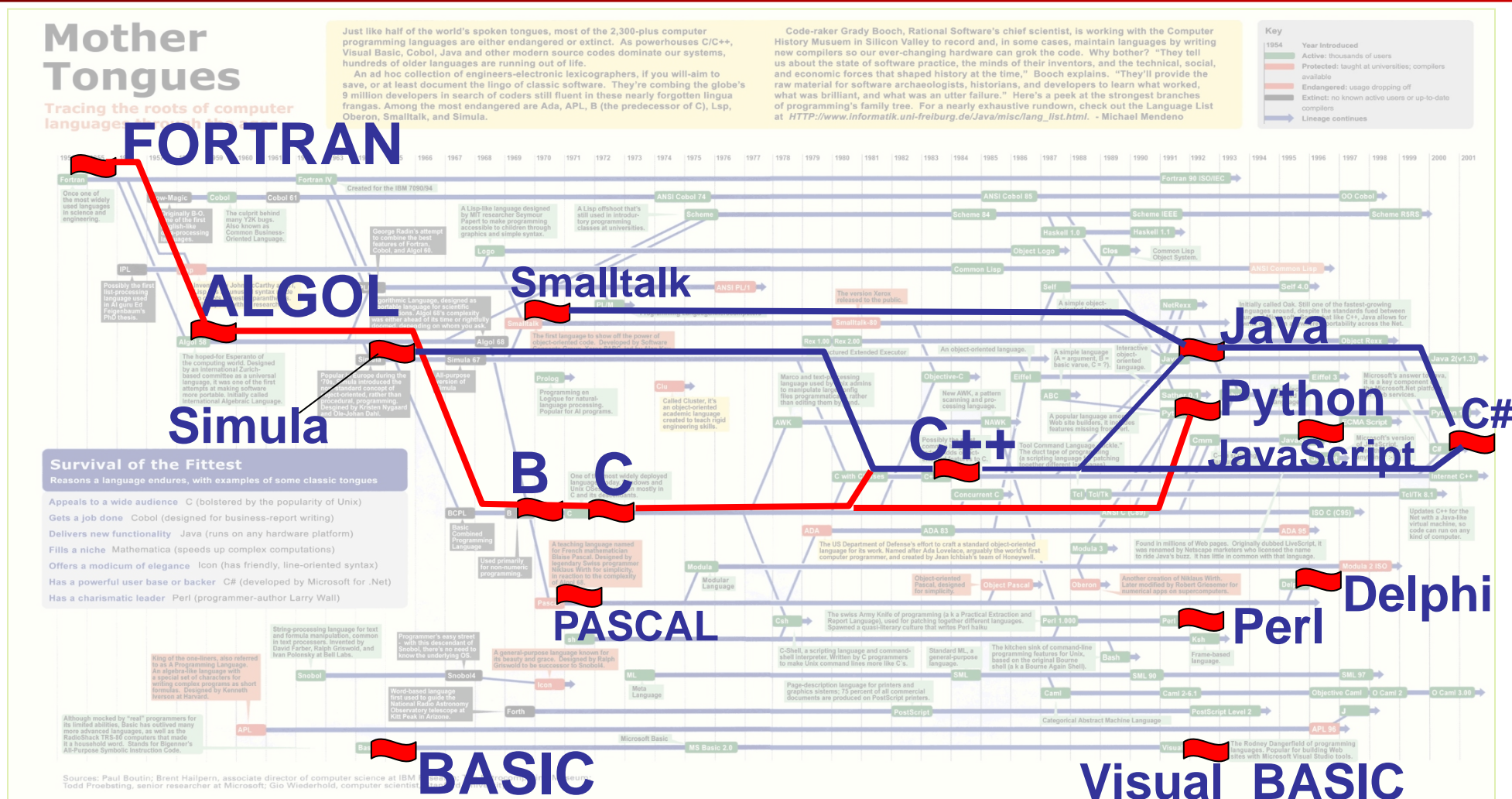
- 1954 Year Introduced
- Active: thousands of users
- Protected: taught at universities; compilers available
- Endangered: usage dropping off
- Extinct: no known active users or up-to-date compilers
- Lineage continues



计算机语言的发展

16

(12) 多种多样的高级语言



不同的计算机语言

示例：功能可能相同，语法格式不同

【问题】比较一下有什么差异？

//C 语言的冒泡排序程序

```
void bubble_sort(int *lists, int count)
{
    int i, j;
    for(i=0; i<count-1; i++)
    {
        for(j=0; j<count-i; j++)
        {
            if (lists[j] < lists[j+1])
            {
                int k = lists[j];
                lists[j] = lists[j+1];
                lists[j+1] = k;
            }
        }
    }
}
```

'Visual Basic 语言的冒泡排序程序

```
Function bubble_sort(lists(1 to 100) as integer, count as integer) As integer
    Dim i as integer, j as integer, k as integer
    for(i=1 to count-1 Step 1)
        for(j=1 to count-i Step 1)
            if (lists(j) < lists(j+1))
                k = lists(j);
                lists(j) = lists(j+1);
                lists(j+1) = k;
            end if
        next j
    next i
End Function
```

Python 语言的冒泡排序程序

```
def bubble_sort(lists, count):
    for i in range(0, count):
        for j in range(0, count-i):
            if lists[j] < lists[j+1]:
                k=lists[j];
                lists[j]=lists[j+1];
                lists[j+1]=k;
    return lists
```

第7讲 程序编写与计算机语言

18

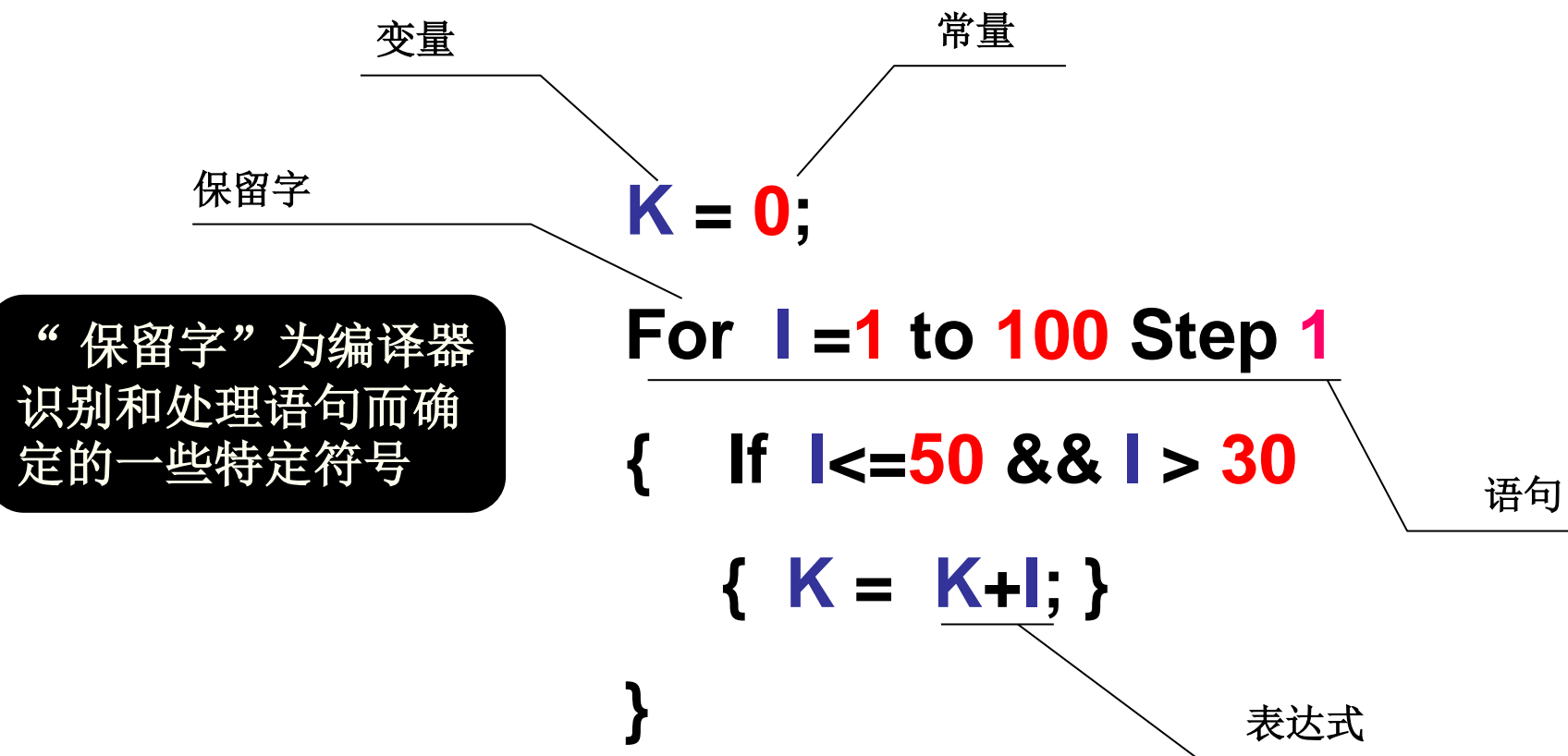
- 一、计算机语言的发展
- 二、高级语言（程序）的基本构成要素
- 三、不同的计算机语言之异同点
- 四、用高级语言构造程序
- 五、计算机语言与编译器

高级语言(程序)的基本构成要素

19

(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



高级语言(程序)的基本构成要素

20

(2)常量、变量与表达式: 你能够书写三种形式的表达式吗?

◆算术表达式示例。算术表达式的结果是一数值;

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

$(B2 + yy4) / L3 - xx3$

◆比较表达式示例。比较表达式的计算结果是逻辑“真”或“假”;

$Grade < 90$

$Grade \geq 70$

$N4 < A1 + B2 + 20$ //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”;

$(x1 \geq A1) \&\& (B2 \lt y2)$

◆将表达式的计算结果赋值给一变量: 赋值语句

$M = X > Y + 50;$

$M = (X > Y) \text{ AND } (X < Y);$

$K = K + (5 * K);$

高级语言(程序)的基本构成要素

21

(3)语句与程序控制：顺序结构？

◆顺序结构

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;
```

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

G5	1
G6	2
G7	3
G8	4
G9	15

高级语言(程序)的基本构成要素

22

(4)语句与程序控制：分支结构？



◆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; }
```

高级语言(程序)的基本构成要素

23

(4)语句与程序控制：分支结构？



X	30
Y	50
Z	80

```
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; }
```

高级语言(程序)的基本构成要素

24

(5)语句与程序控制：循环结构？

◆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	05
I	1

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


高级语言(程序)的基本构成要素

25

(5)语句与程序控制：循环结构？

◆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	1
Sum	01

X=1;

Y=2;

Sum=0;

Do {

Sum = X+Y;

X=X+1;

Y=Y+1;

} While (Sum<=10)

//其他语句

高级语言(程序)的基本构成要素

26

(5)语句与程序控制：循环结构？

◆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	2
Y	2
Sum	0

X=1;

Y=2;

Sum=0;

Do {

Sum = X+Y;

X=X+1;

Y=Y+1;

} While (Sum<0)

//其他语句

高级语言(程序)的基本构成要素

27

(5)语句与程序控制：循环结构？

◆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

X=1;

Y=2;

Sum=0;

While (Sum<0)

Do {

Sum = X+Y;

X=X+1;

Y=Y+1;

}

<其他语句>

高级语言(程序)的基本构成要素

28

(6)高级语言的变量和机器内存的存储单元

高级语言的变量

<pre>Int X=23; Char Y='AB'; Char Z='ABCD';</pre>	变量名	变量值
	<i>x</i>	00000000 00010111
	<i>y</i>	01000001 01000010
	<i>Z</i>	01000001 01000010
		01000011 01000100
		00000000

内存中的存储单元

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	

编译器将不同的变量
映射为存储单元

高级语言(程序)的基本构成要素

29

(7)指针变量是什么?

变量及其存储

		存储地址	存储内容
String *P="ABCD";	P	00000000 00000000	00000100 00001000
		00000000 00000001	00000000 00000100
		00000000 00000010	00001100 00001010
		00000000 00000011	00000000 00000000
String v = "ABCD";	v	00000000 00000100	01000001 01000010
	*p	00000000 00000101	01000011 01000100
		00000000 00000110	00000000 00000000
		00000000 00000111	01000001 01000010
		00000000 00001000	01000011 01000100
		00000000 00001001	00000000 00000000

高级语言(程序)的基本构成要素

30

(8)变量为什么需要声明类型?

“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	00000000 00000000 00000000 00000001 00000000 00000010 00000000 00000011	(注：可通过赋值发生改变)
Sum	00000000 00000100 00000000 00000101	(注：可通过赋值发生改变)
Distance	00000000 00000110 00000000 00000111	(注：可通过赋值发生改变)

高级语言(程序)的基本构成要素

31

(9)如何控制读取向量/数组型变量的不同元素?

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。

编写求上述数组中值的平均值的程序

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

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

向量实例

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

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)

高级语言(程序)的基本构成要素

32

(9)如何控制读取向量/数组型变量的不同元素?

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。

		列			
表实例		1	2	3	4
行	1	11	25	22	25
	2	45	39	8	44
	3	21	28	0	100
	4	34	83	75	16

M[2,3]

```
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;
```

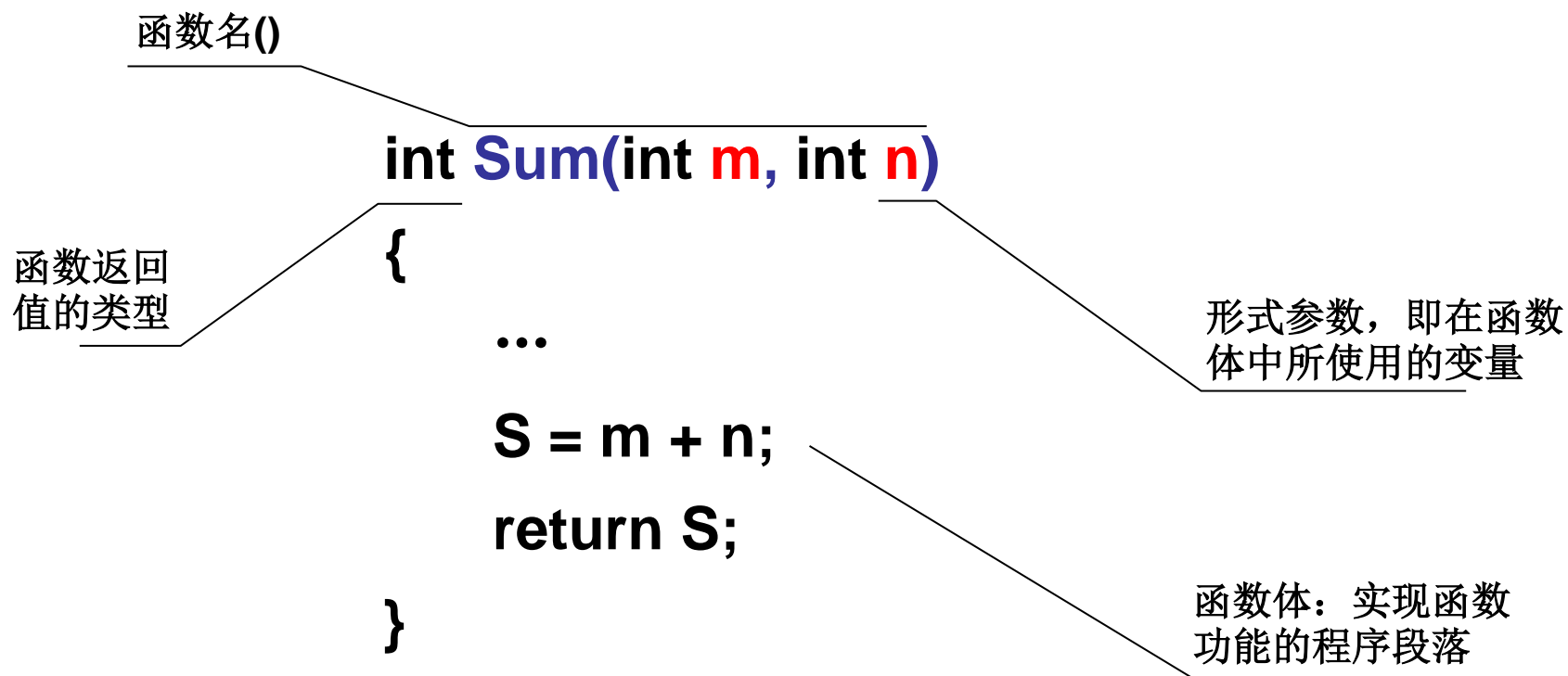
逻辑上是二维的按行、列下标来操作一个元素，如M[2,3]或M[2][3]；物理上仍旧是一维存储的，由“表起始地址+ (行下标-1)*列数+(列下标-1)”。这种转换可由系统自动完成，程序中只需按下标操作即可，即如M[2][3]

高级语言(程序)的基本构成要素

33

(10)函数是很重要的程序构造手段，你知道吗？

函数

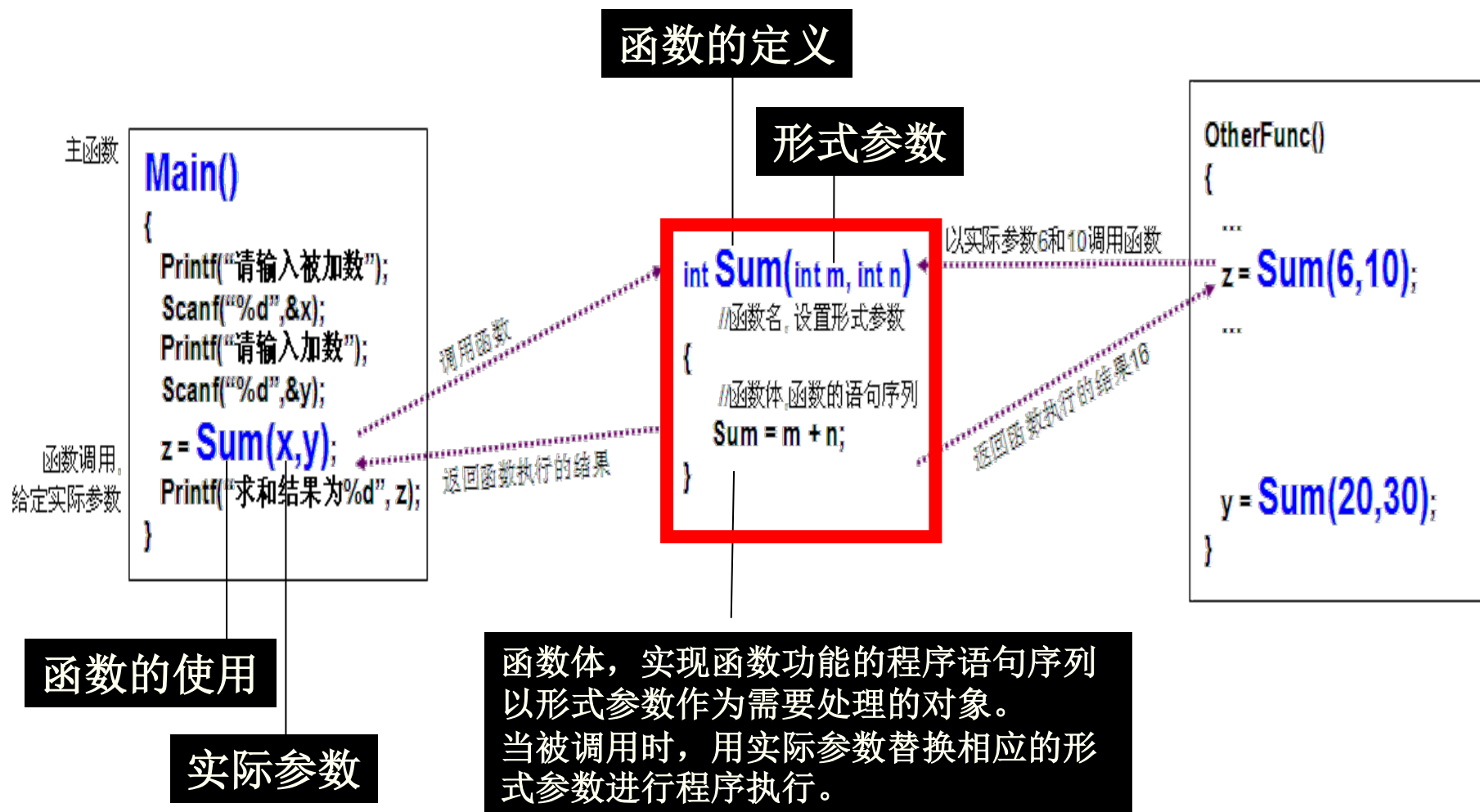


数学上的函数只是一个符号表达，而计算机程序中的函数则是一段可以执行的程序

高级语言(程序)的基本构成要素

34

(11)你知道函数是一种抽象吗？



高级语言(程序)的基本构成要素

35

(12)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

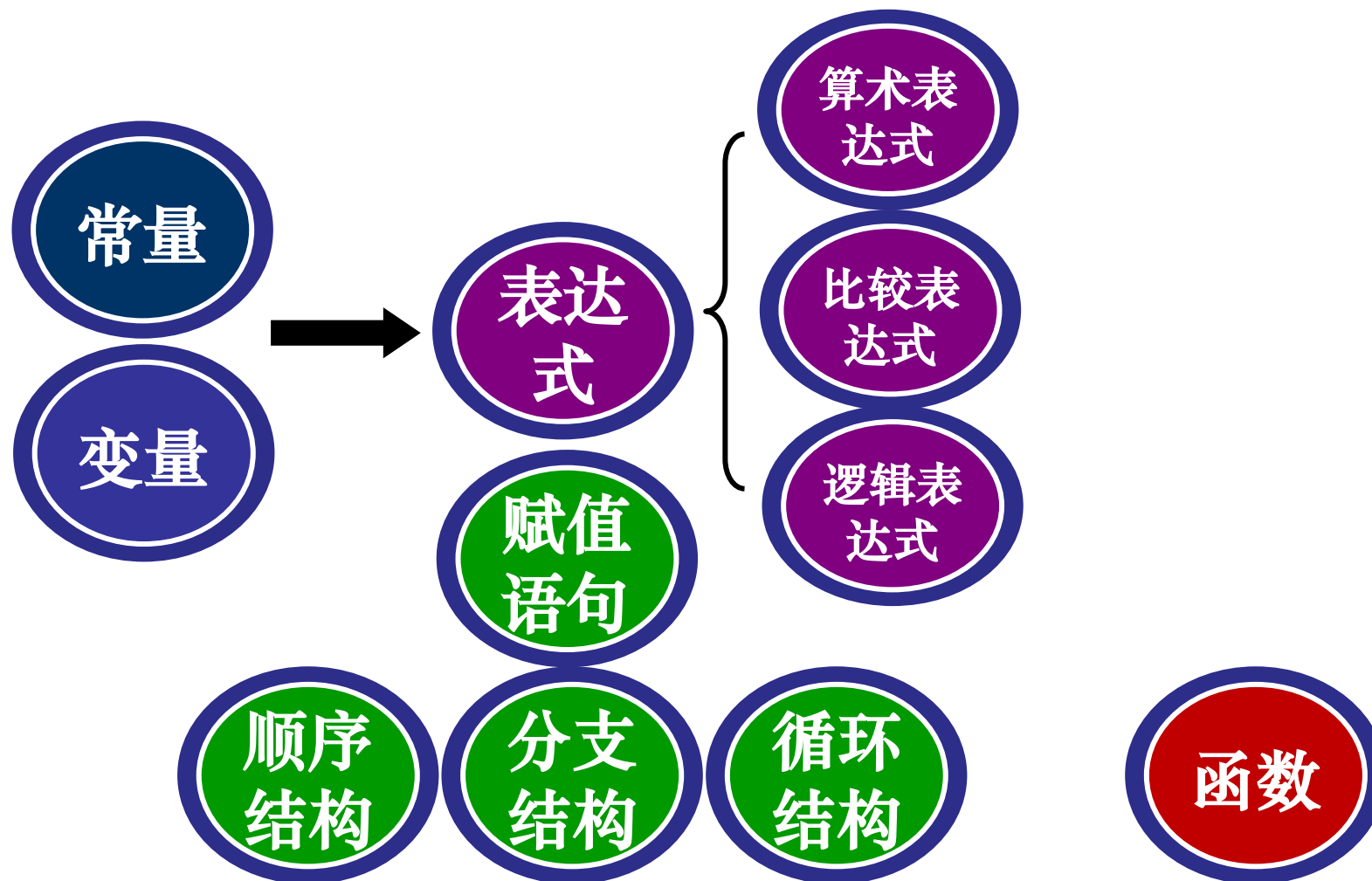
- **数学运算函数**，如三角函数、指数与对数函数、开方函数等；例如 $\sin(\alpha)$ ， $\text{Log}(x)$ 等；
- **数据转换函数**，如字母大小写变换、数值型数字和字符型数字相互转换等；
- **字符串操作函数**，如取子串、计算字符串长度等；例如， $\text{Len}(\text{"abcd"})$ ；
- **输入输出函数**，如输入输出数值、字符、字符串等；例如， $\text{Printf}(\cdots)$ ， $\text{Scanf}(\cdots)$ 等；
- **文件操作函数**，如文件的打开、读取、写入、关闭等；
- **其它函数**，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

高级语言(程序)的基本构成要素

36

(13)小结?

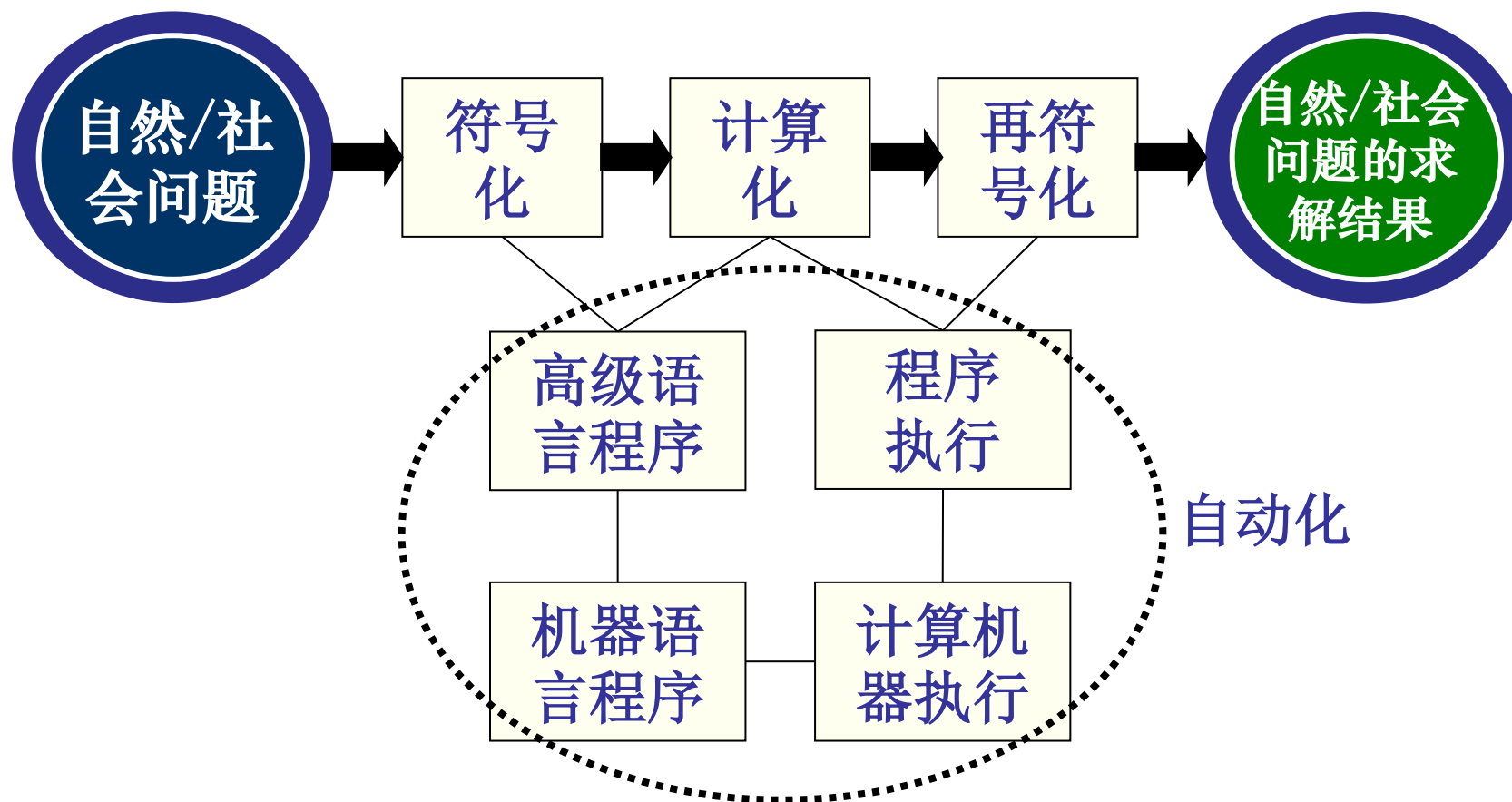
高级语言



高级语言(程序)的基本构成要素

37

(13)小结?



不同的计算机语言之异同点

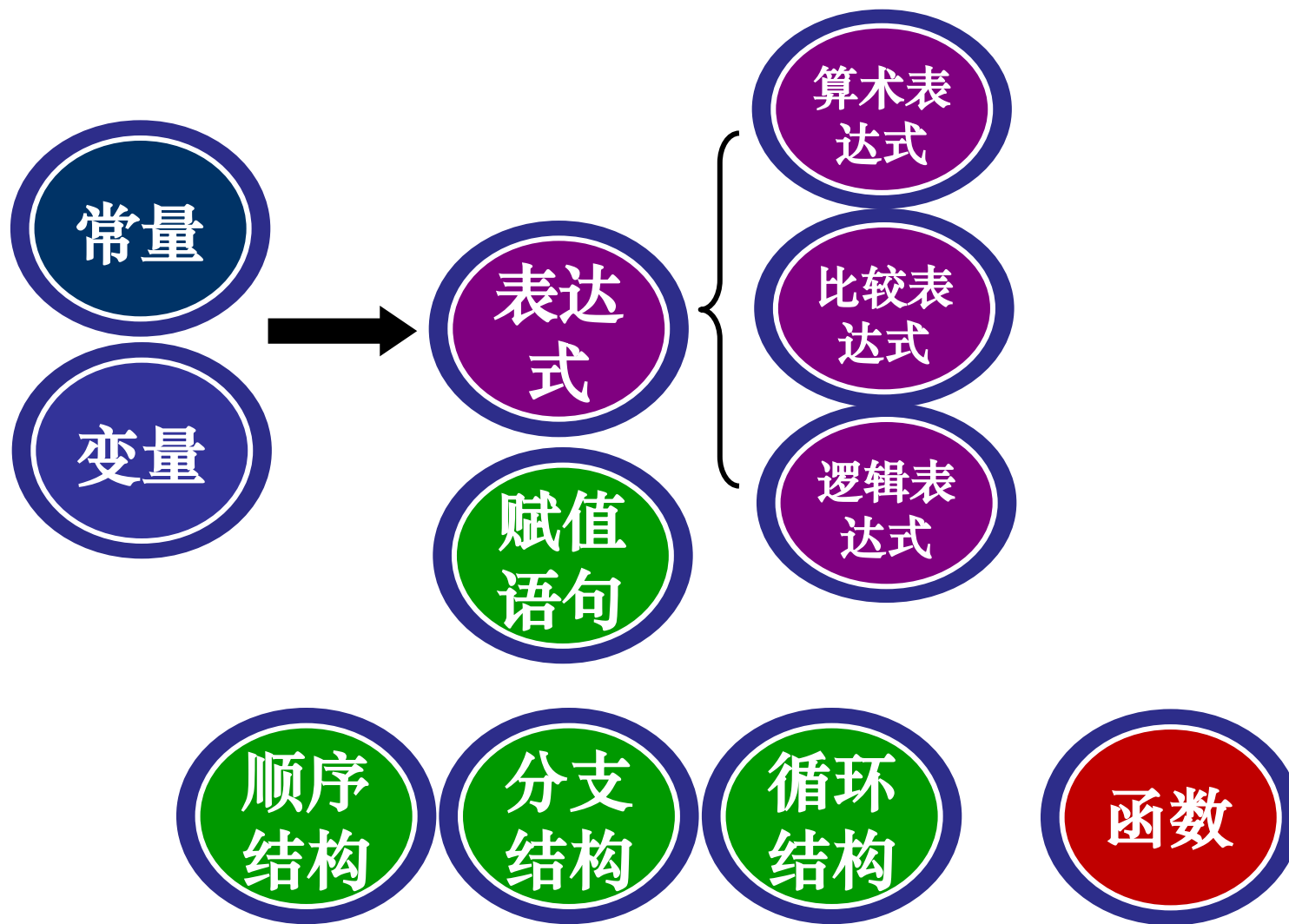
38

//C 语言的冒泡排序程序

```
void bubble_sort(int *lists, int count)
{
    int i, j;
    for(i=0; i<count-1; i++)
    {
        for(j=0; j<count-i; j++)
        {
            if (lists[j] < lists[j+1])
            {
                int k = lists[j];
                lists[j] = lists[j+1];
                lists[j+1] = k;
            }
        }
    }
}
```

Python 语言的冒泡排序程序

```
def bubble_sort(lists, count):
    for i in range(0, count):
        for j in range(0, count-i):
            if lists[j] < lists[j+1]:
                k=lists[j];
                lists[j]=lists[j+1];
                lists[j+1]=k;
    return lists
```



第7讲 程序编写与计算机语言

39

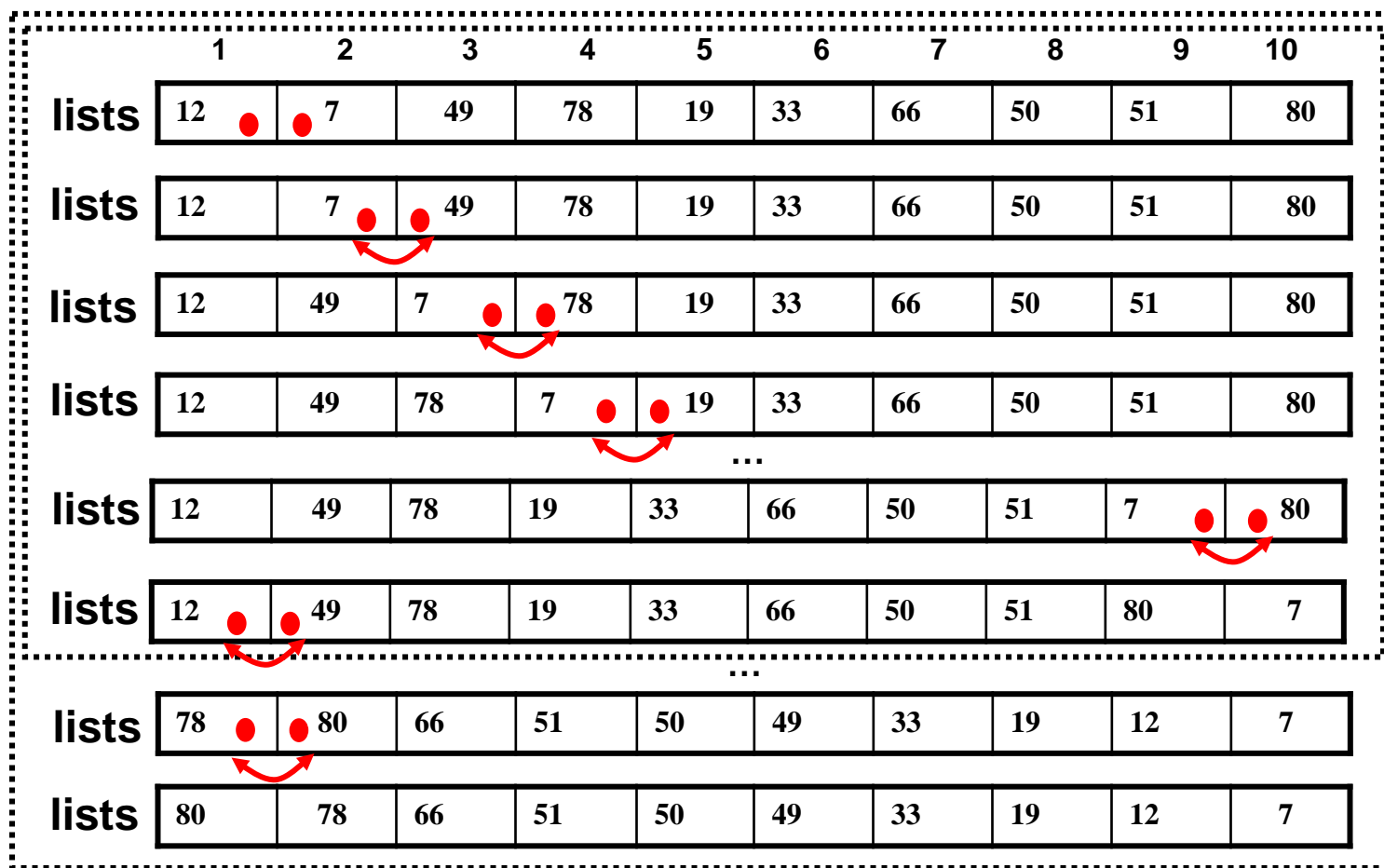
- 一、计算机语言的发展
- 二、高级语言（程序）的基本构成要素
- 三、不同的计算机语言之异同点
- 四、用高级语言构造程序
- 五、计算机语言与编译器

不同的计算机语言之异同点

40

程序编写示例：（1）问题求解思路

编写程序：对一组数据{12, 7, 49, 78, 19, 33, 66, 50, 51, 80}进行由大到小的排序。

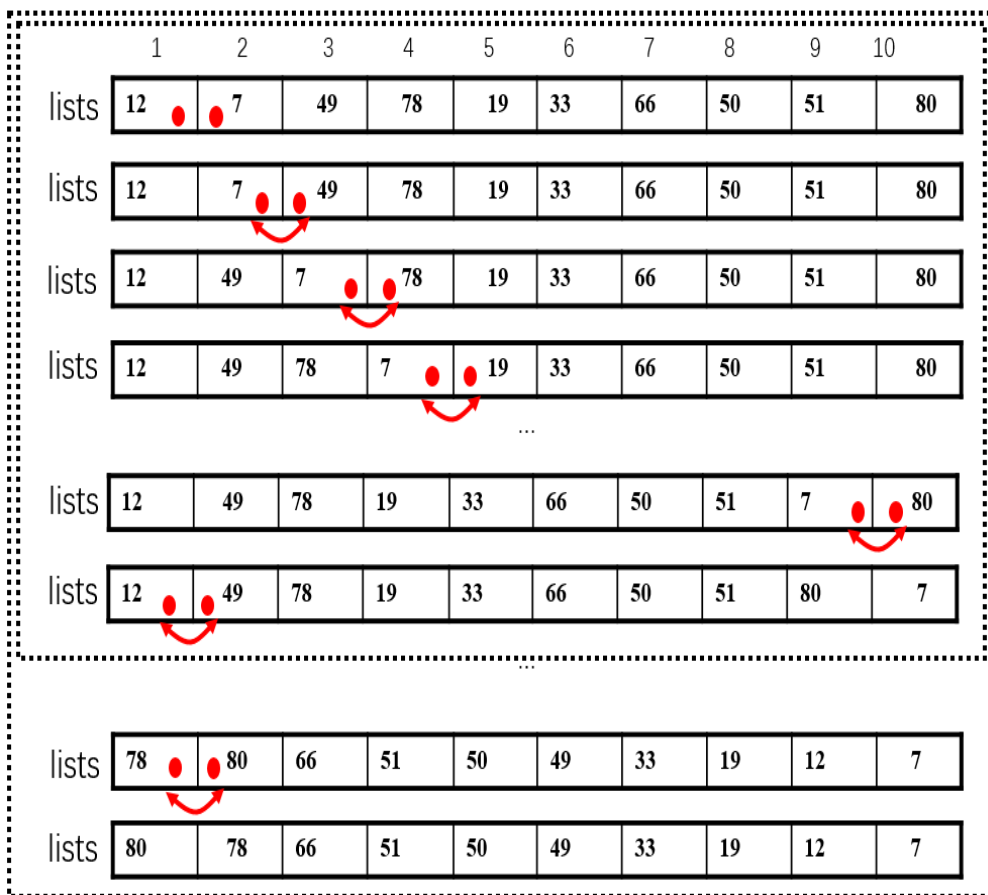


不同的计算机语言之异同点

41

程序编写示例：（2）数据保存

编写程序：对一组数据{12, 7, 49, 78, 19, 33, 66, 50, 51, 80}进行由大到小的排序。



lists[]

lists[0], lists[1], ..., lists[Count-1]

是否 lists[j-1] < lists[j]

j = 1, ..., Count-1

i从1至Count-1 : 控制轮次

j从1至Count-i : 控制每一轮的数的比较

不同的计算机语言之异同点

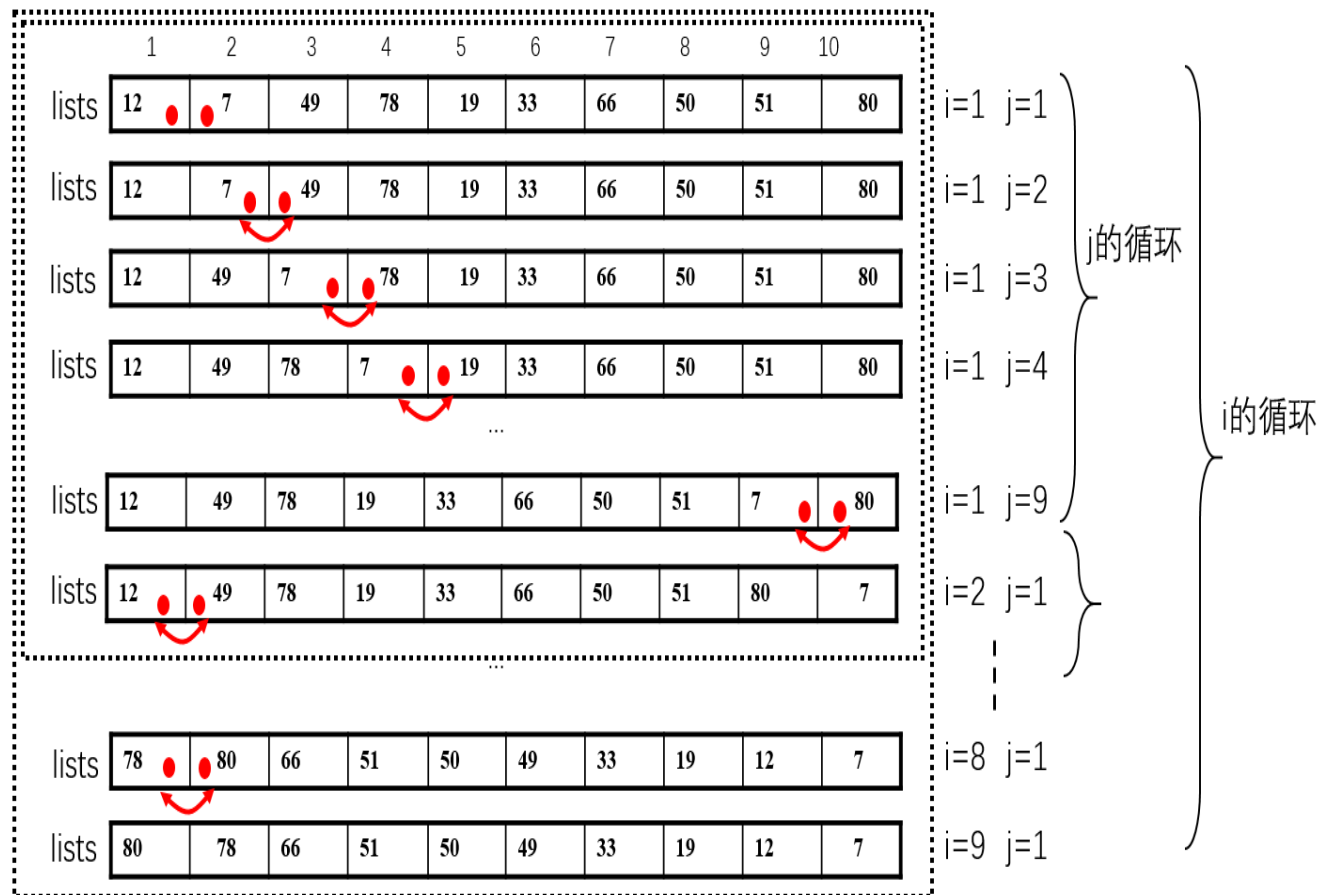
42

程序编写示例：（3）程序表达

编写程序：对一组数据{12, 7, 49, 78, 19, 33, 66, 50, 51, 80}进行由大到小的排序。

```
int Bubble_sort ( int lists[ ], int Count)
```

```
{  
  for i=1 to Count-1  
  {  
    for j=1 to Count-i  
    {  
      if ( lists[ j-1 ] < lists[ j ] ) then  
      {  
        k = lists[ j-1 ];  
        lists[ j-1 ] = lists[ j ];  
        lists[ j ] = k;  
      }  
    }  
  }  
}
```



不同的计算机语言之异同点

43

程序编写示例：（4）不同计算机语言的程序表达

比较一下有什么差异？

//C 语言的冒泡排序程序

```
void bubble_sort(int *lists, int count)
{
    int i, j;
    for(i=0; i<count-1; i++)
    {
        for(j=0; j<count-i; j++)
        {
            if (lists[j] < lists[j+1])
            {
                int k = lists[j];
                lists[j] = lists[j+1];
                lists[j+1] = k;
            }
        }
    }
}
```

要素是
相同的

'Visual Basic 语言的冒泡排序程序

```
Function bubble_sort(lists(1 to 100) as integer, count as integer) As integer
    Dim i as integer, j as integer, k as integer
    for(i=1 to count-1 Step 1)
        for(j=1 to count-i Step 1)
            if (lists(j) < lists(j+1))
                k = lists(j);
                lists(j) = lists(j+1);
                lists(j+1) = k;
            end if
        next j
    next i
End Function
```

书写格式
略不同

要注意查
询手册

Python 语言的冒泡排序程序

```
def bubble_sort(lists, count):
    for i in range(0, count):
        for j in range(0, count-i):
            if lists[j] < lists[j+1]:
                k=lists[j]
                lists[j]=lists[j+1]
                lists[j+1]=k
    return lists
```

不同的计算机语言之异同点

44

//C 语言的冒泡排序程序

```
void bubble_sort(int *lists, int count)
```

```
{  int i, j;
    for(i=0; i<count-1; i++)
    {  for(j=0; j<count-i; j++)
        {  if  (lists[j] < lists[j+1])
            {  int k = lists[j];
                lists[j] = lists[j+1];
                lists[j+1] = k;
            }
        }
    }
}
```

Python 语言的冒泡排序程序

```
def bubble_sort(lists, count):
```

```
    for i in range(0, count):
        for j in range(0, count-i):
            if lists[j] < lists[j+1]:
                k=lists[j];
                lists[j]=lists[j+1];
                lists[j+1]=k;
    return lists
```

基本程序要素 (本书)	Python	C	Visual Basic
算术表达式 +、-、*、/、**(幂)、mod(取模)	基本部分: 相同 +、-、*、/、**、%(取模)、//(取整)	基本部分: 相同 +、-、*、/、%(取模)	基本部分: 相同 +、-、*、/、^(幂)、mod(取模)
a + b a**3 9 mod 4	a + b a**3 9 % 4	a + b 9 % 4	a + b a^3 9 mod 4
比较表达式/关系运算符 ==、<、>、<=、>=、<=	相同 ==、!=(<>)、>、<、>=、<=	基本部分: 相同 ==、!=、>、<、>=、<=	相同 ==、<、>、<=、>=、<=
a > b a < b	a > b a != b	a > b a != b	a > b a < b
逻辑运算符 and、or、not	相同 and、or、not	不同 &&、 、!	相同 and、or、not
(a>b) and (c>d)	(a>b) and (c>d)	(a>b) && (c>d)	(a>b) and (c>d)
赋值语句 数据类型 变量名; 变量 = 表达式;	相同: 可直接赋值 变量 = 表达式	相同: 变量需事先声明 数据类型 变量名; 变量 = 表达式;	相同: 变量需事先声明 Dim 变量名 AS 类型 变量 = 表达式;
int i, j; i = a + b;	i = a + b	int i; i = a + b	Dim i as integer i = a + b
If 条件 Then { 执行语句 11; 执行语句 12; } Else { 执行语句 2;}	If 条件: 执行语句 11 执行语句 12 Else: 执行语句 2 (注: 用等长缩进区分语句块)	If 条件 Then { 执行语句 11; 执行语句 12; } Else { 执行语句 2;}	If 条件 Then 执行语句 11; 执行语句 12; Else 执行语句 2 EndIf
If (a>b) Then { c=a; a=b; } Else { b=a; a=c; }	If a>b: c=a a=b Else b=a a=c	If (a>b) Then { c=a; a=b; } Else { b=a; a=c; }	If (a>b) Then c=a a=b Else b=a; a=c; EndIf
For 计数器=起始值 to 结束值 { 循环语句 1; 循环语句 2; }	for 计数器 in range(下界值, 上界值): 循环语句 1 循环语句 2 (注: 用等长缩进区分语句块)	For(计数器=初值; 条件; 更新) { 循环语句 1; 循环语句 2; }	For 计数器 = 起始值 to 结束值 循环语句 1; 循环语句 2; Next 计数器

不同的计算机语言之异同点

45

```
void bubble_sort(int *lists, int count)
{
    int i, j;
    for(i=0; i<count-1; i++)
    {
        for(j=0; j<count-i; j++)
        {
            if (lists[j] < lists[j+1])
            {
                int k = lists[j];
                lists[j] = lists[j+1];
                lists[j+1] = k;
            }
        }
    }
}
```

Python 语言的冒泡排序程序

```
def bubble_sort(lists, count):
    for i in range(0, count):
        for j in range(0, count-i):
            if lists[j] < lists[j+1]:
                k=lists[j]
                lists[j]=lists[j+1]
                lists[j+1]=k
    return lists
```

基本程序要素 (本书)	Python	C	Visual Basic
For a = 10 to 20 Step 1 { Sum = Sum + a; }	For a in range(10, 20): Sum = Sum + a	For (int a = 10; a < 20; a = a + 1) { Sum = Sum + a; }	For a = 10 to 20 Step 1 Sum = Sum + a; Next a
While (条件) { 循环语句 1; 循环语句 2; }	While 条件: 循环语句 1 循环语句 2 (注: 用等长缩进区分语句块)	While (条件) { 循环语句 1; 循环语句 2; }	Do While 条件 循环语句 1; 循环语句 2; Loop
While (a < 20) { Sum = Sum + a; a = a - 1; }	While a < 20: Sum = Sum + a a = a - 1	While (a < 20) { Sum = Sum + a; a = a - 1; }	Do While (a < 20) Sum = Sum + a a = a - 1 Loop
类型 函数名(Para) { 函数体; } 变量 = funcname(para)	def 函数名(para): 函数体 return [返回值] (注: 用等长缩进区分语句块) 变量 = funcname(para)	类型 函数名(Para) { 函数体; } 变量 = funcname(para)	Function 函数名(Para) As 类型 函数体 End Function 变量 = funcname(para)
Long Fib(int n) { Fib = 120; } ret = Fib(5);	def Fib(n): Fib = 120 ret = Fib(5)	Long Fib(int n) { Fib = 120; } ret = Fib(5);	Function Fib(n As Integer) As Long Fib = 120 End Function ret = Fib(5)

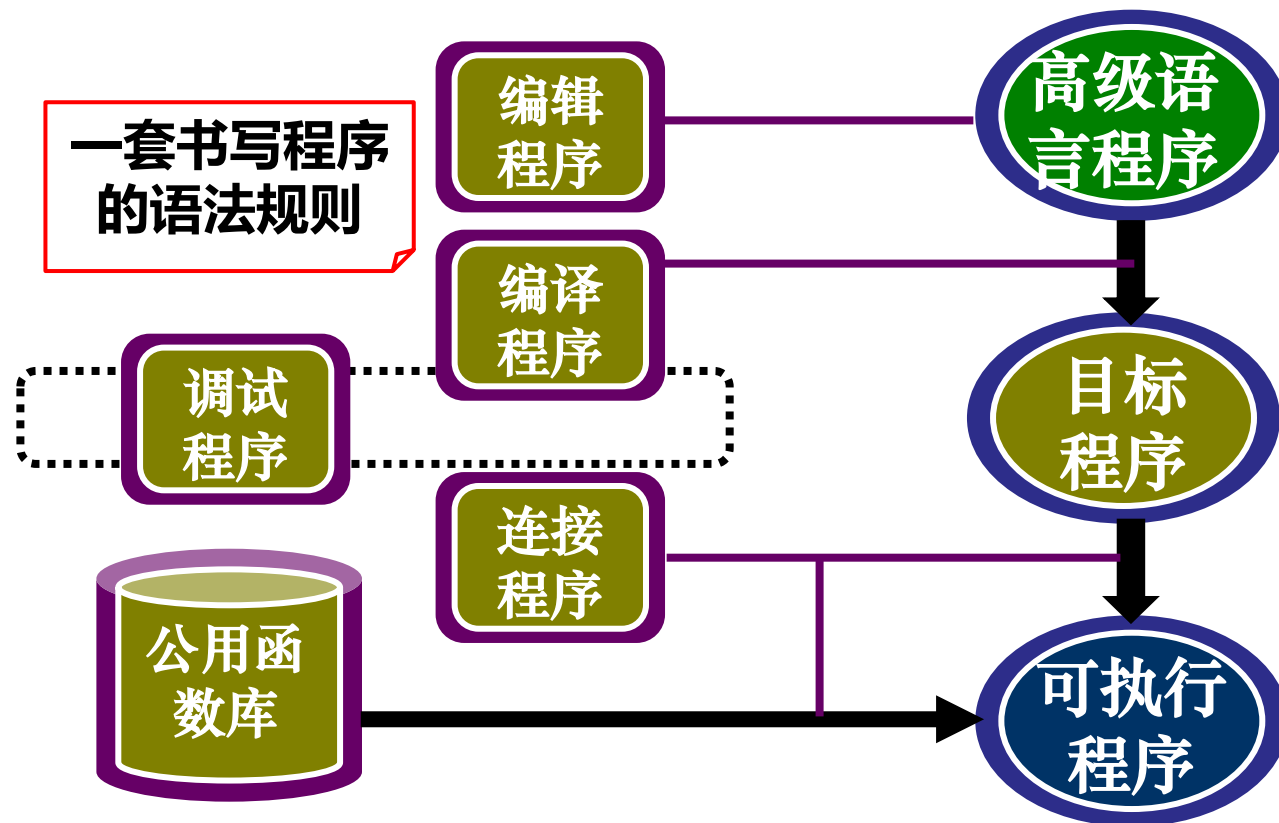
不同的计算机语言之异同点

46

程序设计过程与程序设计环境

程序设计过程：编辑源程序→编译→连接→执行。

计算机语言程序设计环境：
编辑、编译、连接、
调试、运行一体化平台



第7讲 程序编写与计算机语言

47

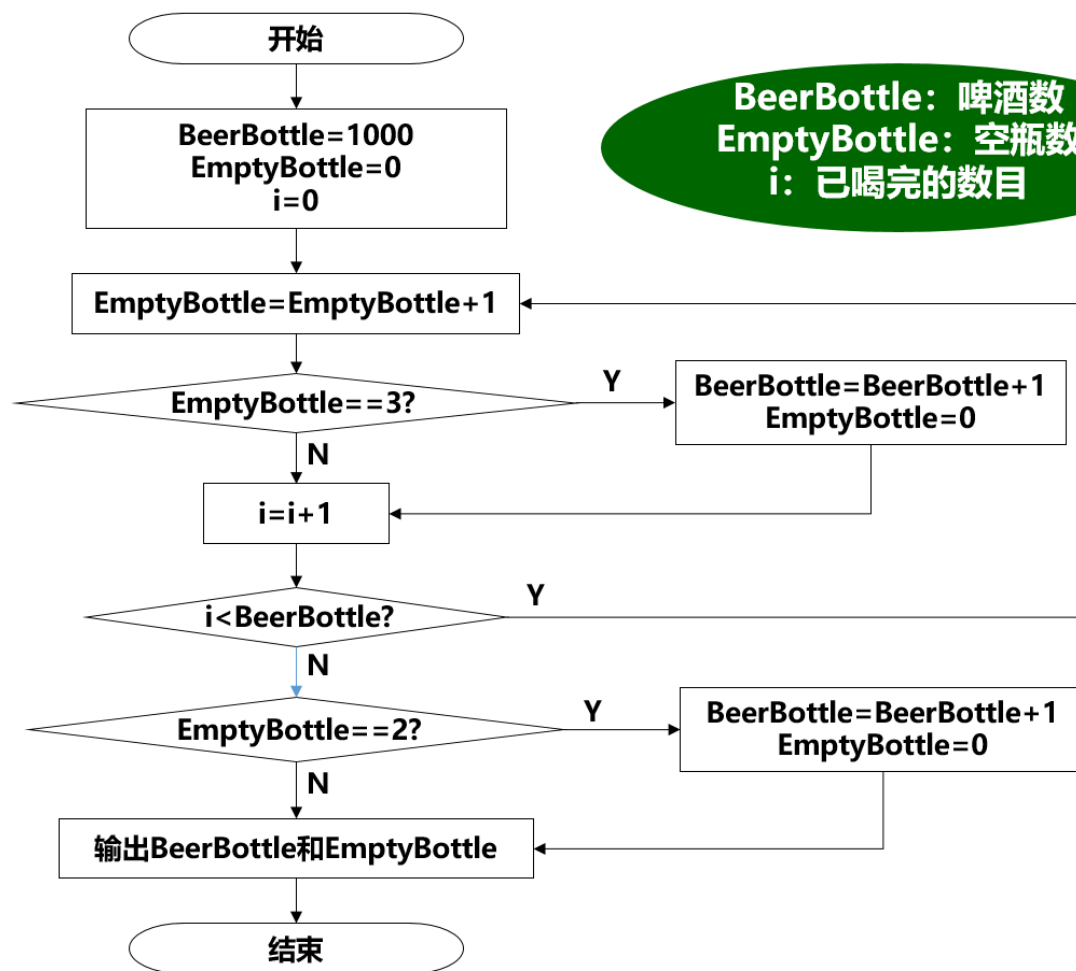
- 一、计算机语言的发展
- 二、高级语言（程序）的基本构成要素
- 三、不同的计算机语言之异同点
- 四、用高级语言构造程序
- 五、计算机语言与编译器

用高级语言构造程序

48

(1) 程序构造示例

【示例】有1000瓶啤酒，每喝完一瓶得到一个空瓶子，每3个空瓶子又能换1瓶啤酒，喝掉以后又得到一个空瓶子。问总共能喝多少瓶啤酒？还剩多少空瓶子？

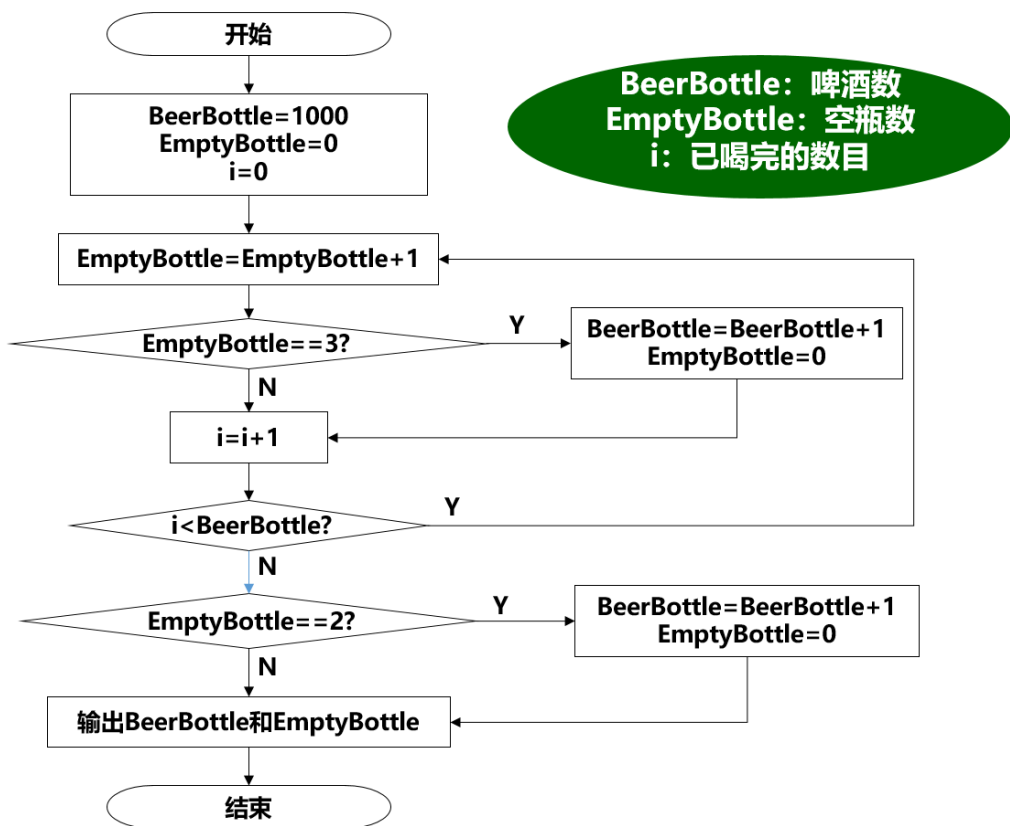


用高级语言构造程序

49

(1) 程序构造示例

【示例】有1000瓶啤酒，每喝完一瓶得到一个空瓶子，每3个空瓶子又能换1瓶啤酒，喝掉以后又得到一个空瓶子。问总共能喝多少瓶啤酒？还剩多少空瓶子？



```
int HowMuchBeer()
{ int BeerBottle = 1000;
  int EmptyBottle = 0;
  for i=0 to BeerBottle
  { EmptyBottle = EmptyBottle+1;
    if (EmptyBottle ==3) then
    { BeerBottle=BeerBottle+1;
      EmptyBottle=0;
    }
  }
  if (EmptyBottle ==2)
  { BeerBottle=BeerBottle+1;
    EmptyBottle=0;
  }
  Printf ("EmptyBottle=%d", EmptyBottle);
  Printf ("BeerBottle=%d", BeerBottle);
}
```

//在最后剩2个空瓶时，加1瓶即可

用高级语言构造程序

50

用不同语言书写程序

【示例】 编写求n的阶乘的程序

【C语言程序】 阶乘的递归程序

```
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*/
}
```

靠匹配大
括号区分
语句块

【Python语言程序】 阶乘的递归程序

```
def Fact (n):
    if (n > 1):
        x = Fact(n - 1);      # 递归调用
        return( n*x);
    else :
        return(1);           # 递归基础
```

靠等长缩
进区分语
句块

用高级语言构造程序

51

用不同语言书写程序

【示例】 编写求n的阶乘的程序

【C语言程序】 阶乘的迭代程序

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

靠匹配大
括号区分
语句块

【Python语言程序】 阶乘的迭代程序

```
def Fact( n):
    product = 1;
    for counter in range(1 , n+1):
        product = product * counter;
    #迭代
    return product;
```

靠等长缩
进区分语
句块

用高级语言构造程序

52

用不同语言书写程序

【示例】编写求解斐波那契数列的程序

【C语言程序】斐波那契数列的递归程序

```
long Fib(int n)
{ if (n == 0 or n == 1)
  { return n;
    /*递归基础*/
  }
  else
  { return Fib(n-1) + Fib(n-2);
    /*递归调用*/
  }
}
```

【C语言程序】斐波那契数列的迭代程序

```
long Fib_Iteration(int n)
{ int X, Y, Z, I;
  if (n==0 or n==1) { Y=0; return Y; }
  else { X=0; Y=1;
    for I=1 to n-1 step 1
    {
      Z=X+Y;
      X=Y;
      Y=Z;
    }
    return Y ;
  }
}
```

靠匹配大
括号区分
语句块

用高级语言构造程序

53

用不同语言书写程序

【示例】编写求解斐波那契数列的程序

【Python语言程序】斐波那契数列的递归程序

```
def Fib(n):  
    if (n == 0 or n == 1):  
        return n;  
        # 递归基础  
    else :  
        return Fib(n-1) + Fib(n-2);  
        # 递归调用
```

【Python语言程序】斐波那契数列的迭代程序

```
def Fib_Iteration(n):  
    if (n==0 or n==1):  
        Y=0;  
        return Y;  
    else :  
        X=0;  
        Y=1;  
        for l in range (1, n):  
            Z=X+Y;  
            X=Y;  
            Y=Z;  
        return Y ;
```

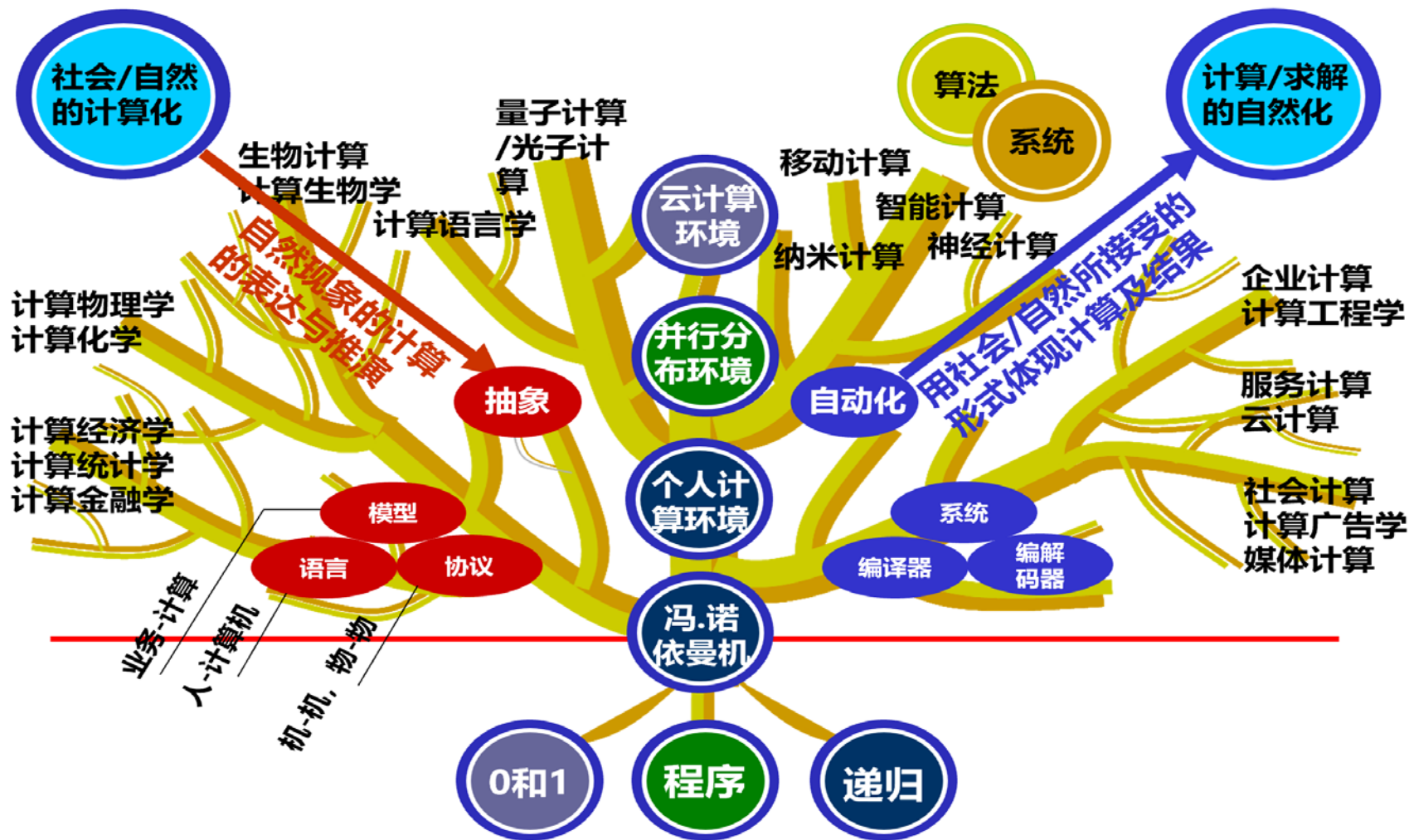
靠等长缩
进区分语
句块

第7讲 程序编写与计算机语言

54

- 一、计算机语言的发展
- 二、高级语言（程序）的基本构成要素
- 三、不同的计算机语言之异同点
- 四、用高级语言构造程序
- 五、计算机语言与编译器

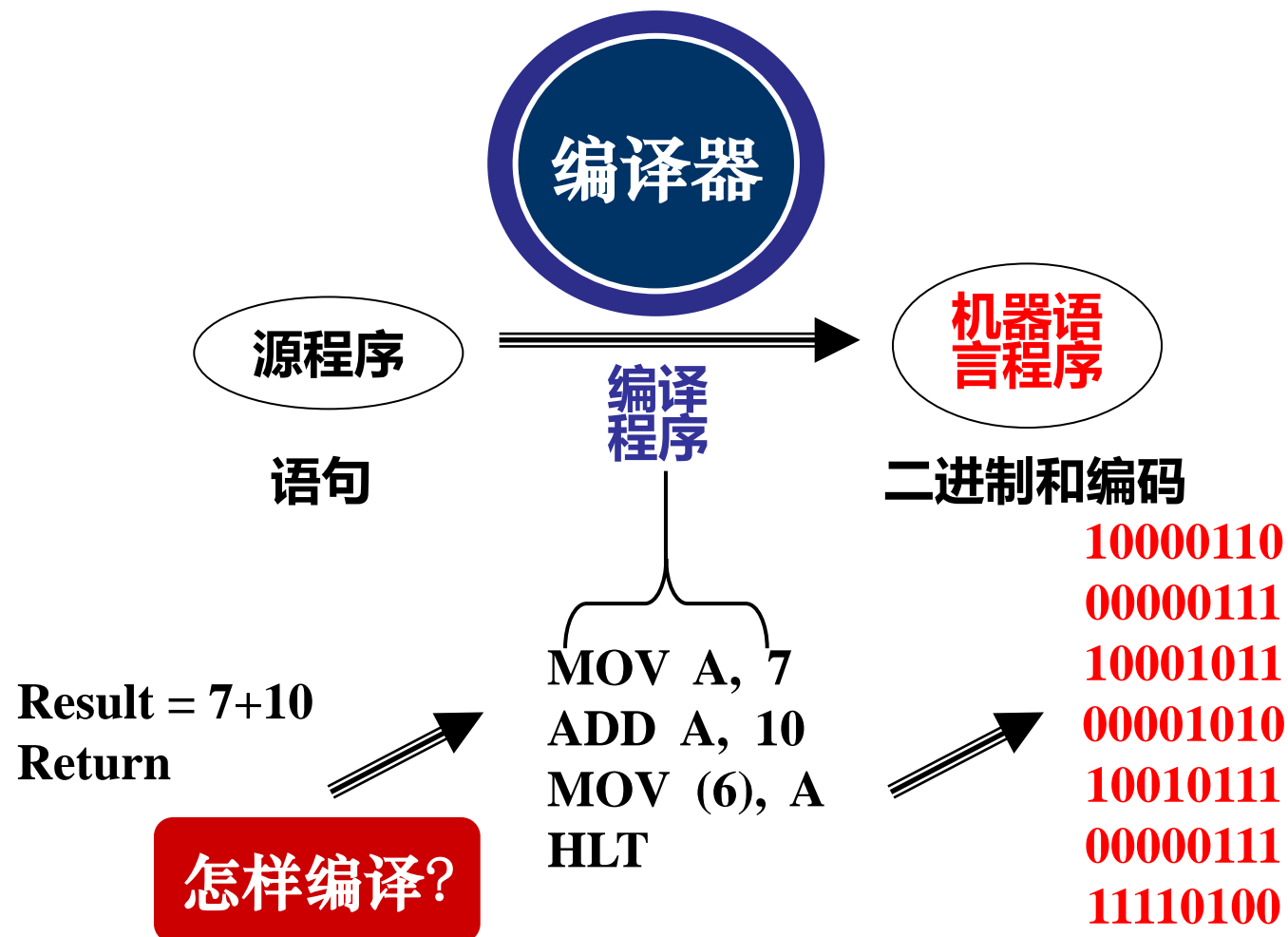
(1)你记得计算之树中的不同抽象层次吗?



计算机语言与编译器--一种抽象-自动化机制示例

56

(2)为什么高级语言程序需要编译?



计算机语言与编译器--一种抽象-自动化机制示例

57

(3)高级语言中的模式化的语句?

由“**具体的**”运算式到“**模式**”运算式

Result = 7 + 10;

Sum = 8 + 15;

K = 100 + 105;

....

V = C + C;

注:

Result: 具体的变量
7, 10: 具体的

= 赋值符号
+ 加法运算符号
; 语句结束符

变化的部分

不变的部分
(保留字)

注:

V: 变量
C: 常量

= 赋值符号
+ 加法运算符号
; 语句结束符

计算机语言与编译器--一种抽象-自动化机制示例

58

(4)语句模式的识别

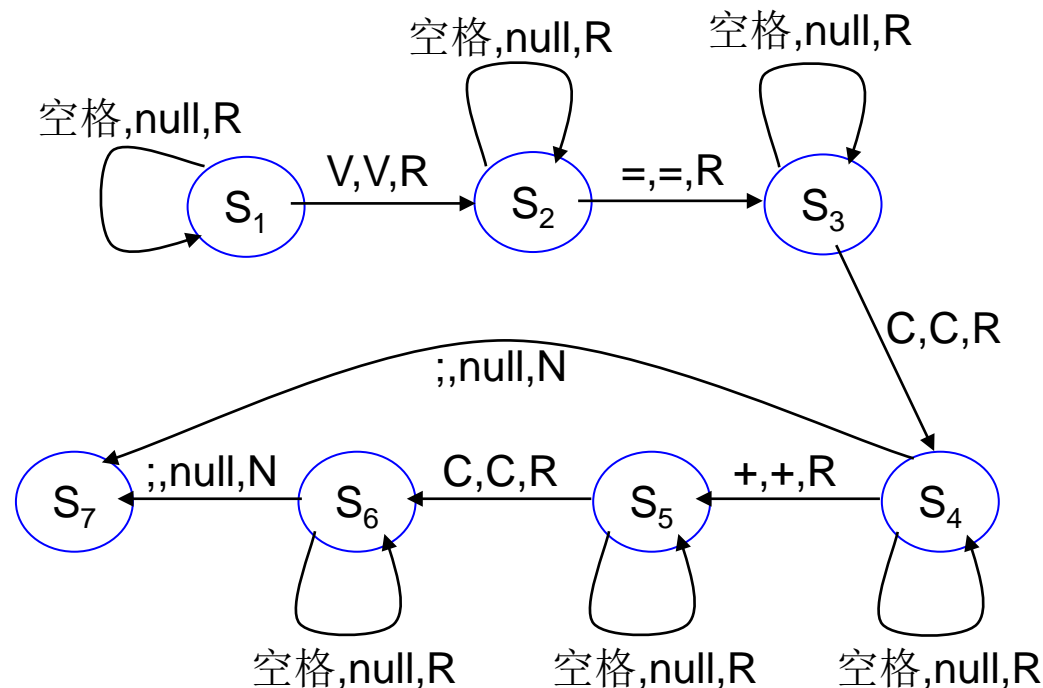
“模式” 运算式的识别
及常量、变量的标识

Result = 7 + 10;



(V, 1) = (C, 1) + (C, 2);

注：字母表{V, C, =, +, 空格, ;}; S₁起始状态; S₇终止状态; null表示什么也不写回。



(c)能识别两种模式 “V=C;”和 “V=C+C;”并能去除空格的图灵机示意图

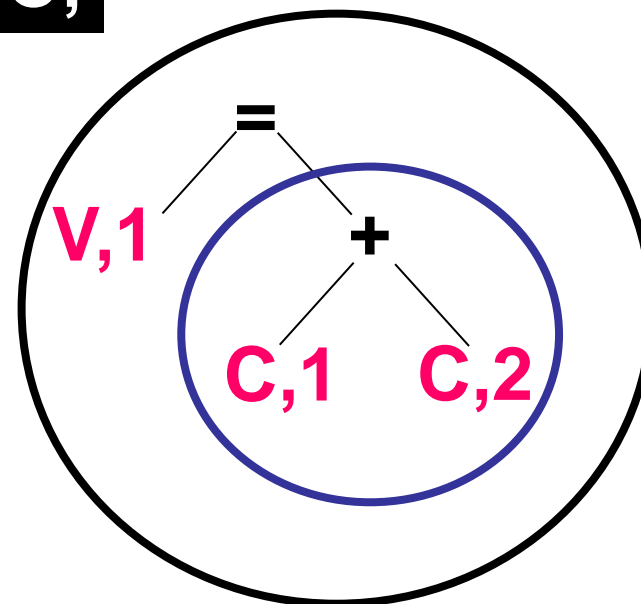
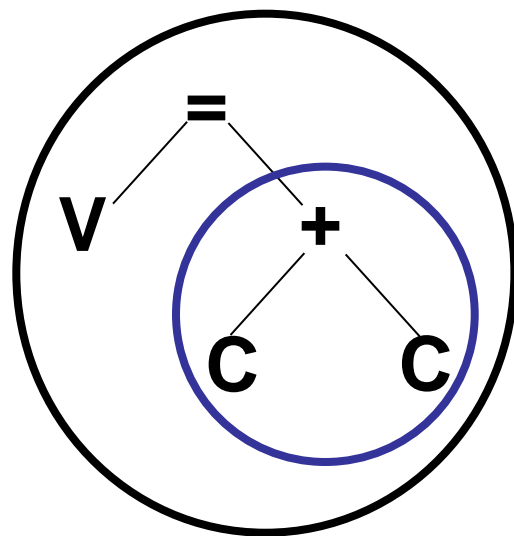
计算机语言与编译器--一种抽象-自动化机制示例

59

(5)复杂模式的预先构造

复杂模式转换为简单模式及其组合

$V = C + C;$

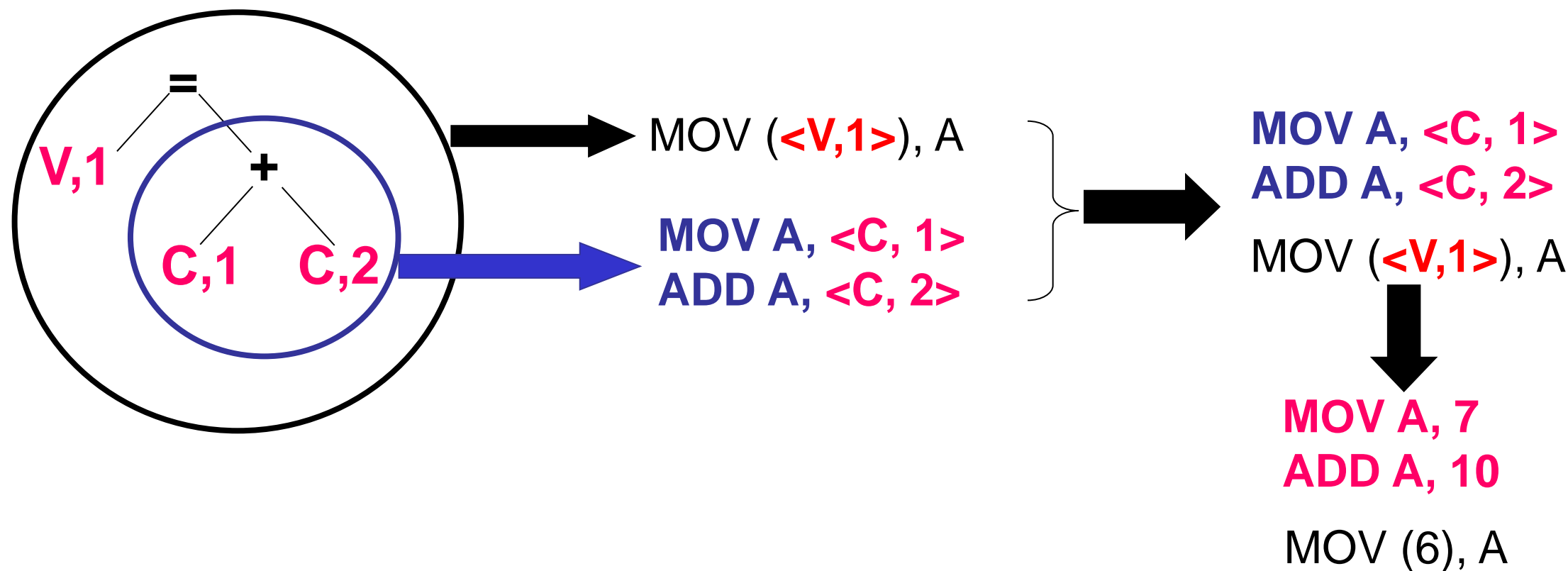


计算机语言与编译器--一种抽象-自动化机制示例

60

(6)简单模式与汇编语句的映射

将简单模式转换成汇编语言语句序列，用常量值和变量地址进行替换，组合次序调整，得到最后的汇编语言程序

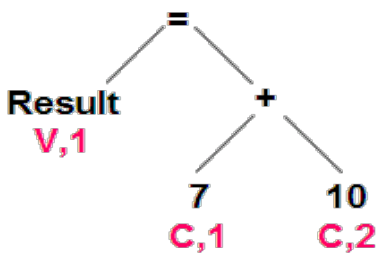


计算机语言与编译器--一种抽象-自动化机制示例

(7)小结

Result = 7 + 10;

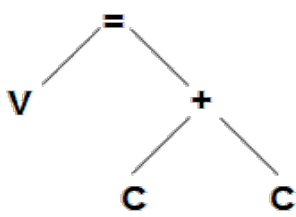
注:
Result: 具体的变量
7, 10: 具体的常量
=: 赋值符号
+: 加法运算符号



(a)一种具体的语句及其解析结构

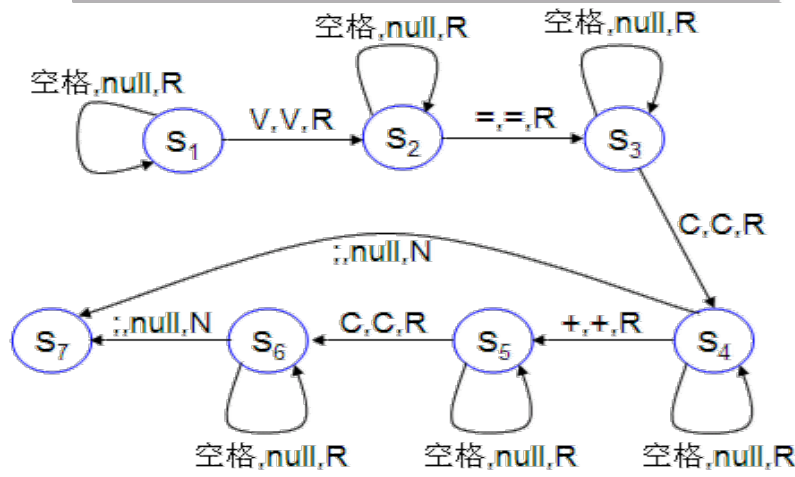
V = C + C;

注:
V: 变量
C: 常量
=: 赋值符号
+: 加法运算符号

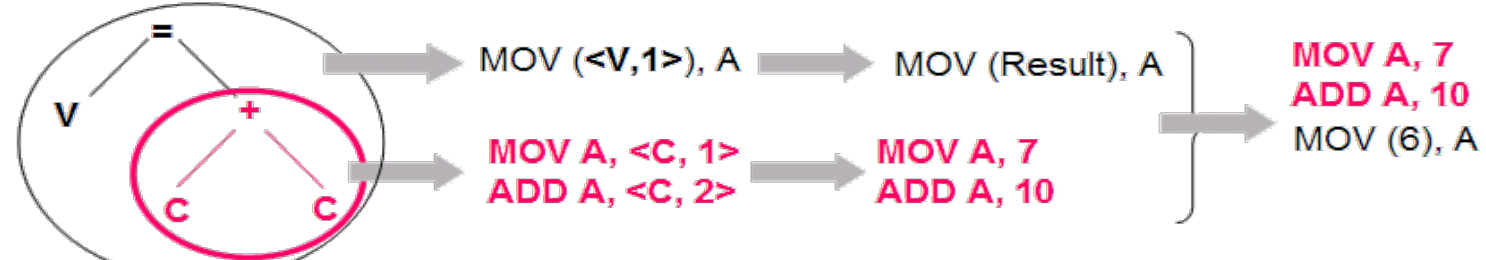


(b)图(a)所示语句的一种模式及其解析结构

注: 字母表{V, C, =, +, 空格, ;}; S₁起始状态; S₇终止状态; null表示什么也不写回。



(c)能识别两种模式“V=C;”和“V=C+C;”并能去除空格的图灵机示意图



(d)语法分析树转换成汇编语言语句的过程示意

计算机语言与编译器--一种抽象-自动化机制示例

(7)小结

