编译原理

田玲 教授、博导

lingtian@uestc.edu.cn

课程安排

课程设置

- 56学时
- 48课堂授课+8学时实验

上课时间与地点：

星期一 1-2节 立人楼B101

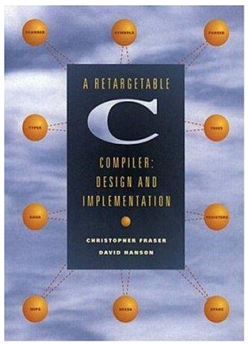
星期三 1-2节 立人楼B105

先修课程

- 《数据结构》、《形式语言》、《C语言》、《汇编语言》

教材和参考书

- 王晓斌、田玲等. 程序设计语言与编译——语言的设计与实现（第4版）
- 蒋宗礼、姜守旭. 形式语言与自动机理论(第2版)
- Even S. Advanced. Compiler Design and Implementation.



课程安排

上课时间与地点：

星期一 1-2节　　立人楼B101

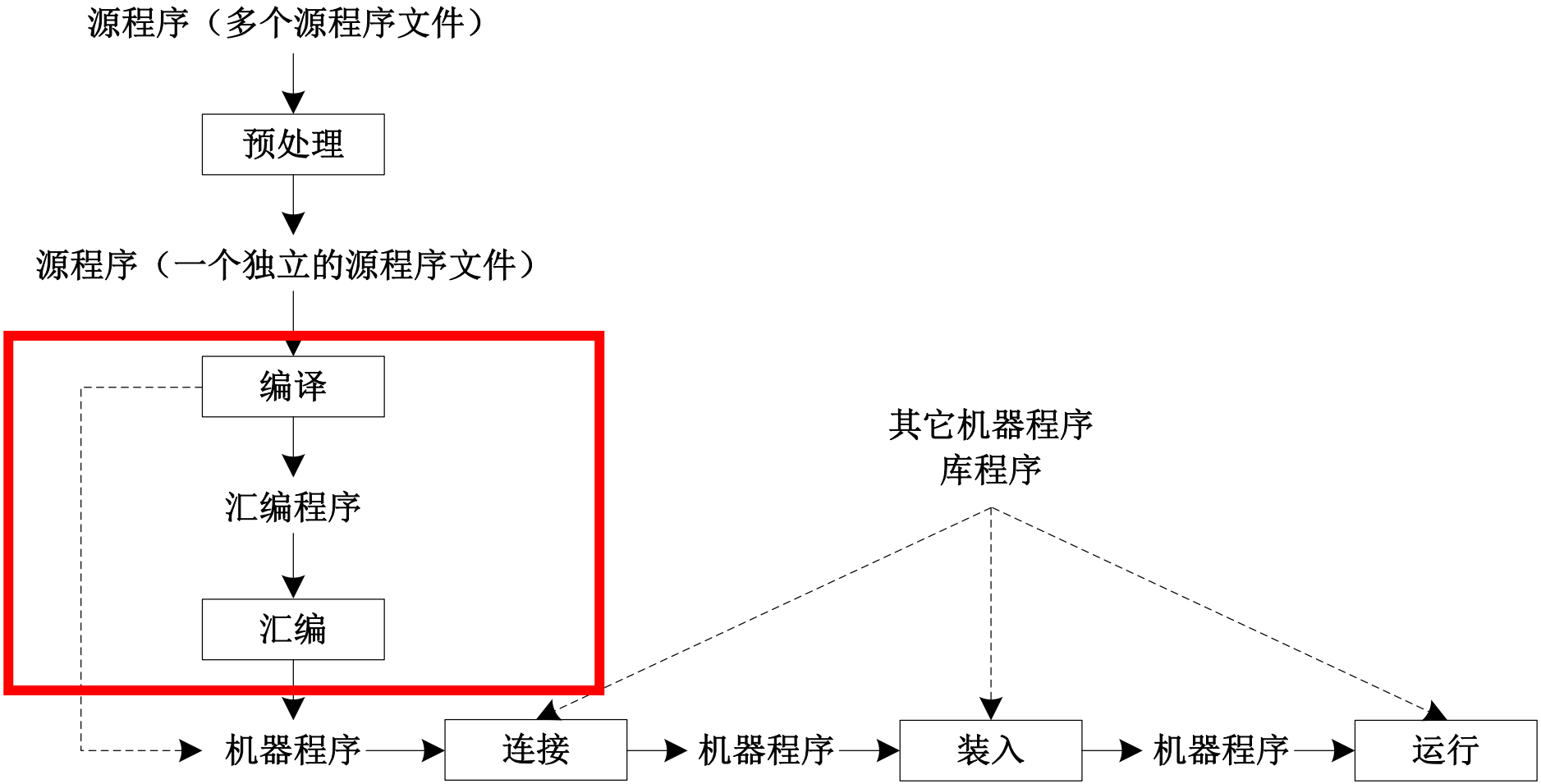
星期三 1-2节　　立人楼B105

答疑：根据情况安排

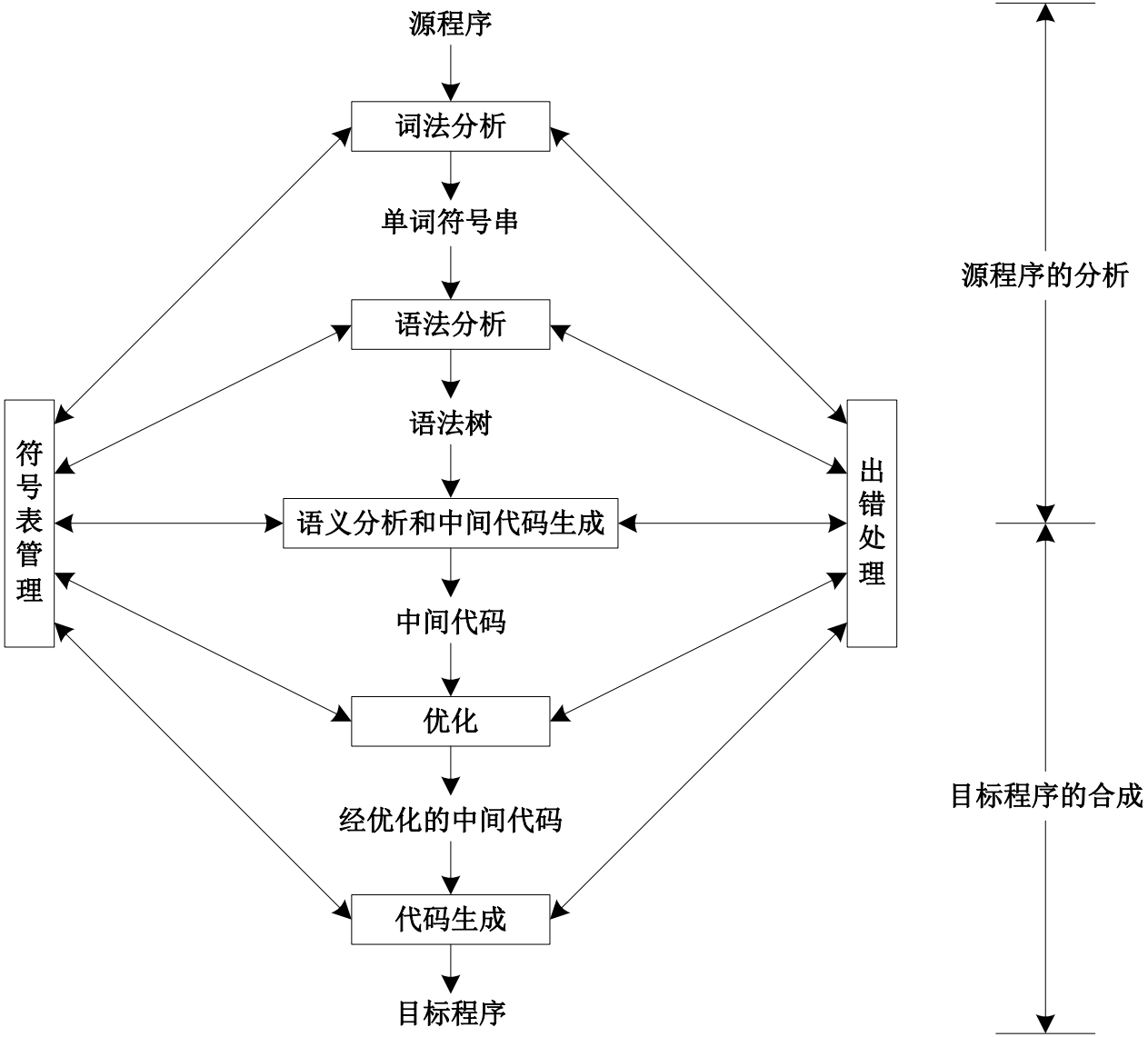
作业：根据进度，布置作业后一周至两周

最后成绩=期终（70%）+半期考试(10%)+考勤及平时作业（10%）+实验（10%）

完整的程序处理过程


源程序（多个源程序文件）
预处理
源程序（一个独立的源程序文件）
编译
汇编程序
汇编
其它机器程序
库程序
机器程序
连接
机器程序
装入
机器程序
运行

编译步骤


源程序
词法分析
单词符号串
语法分析
语法树
语义分析和中间代码生成
中间代码
优化
经优化的中间代码
代码生成
目标程序
符号表管理
出错处理
源程序的分析
目标程序的合成

教学目标

通过本课程的学习，使学生掌握设计和实现一个程序设计语言的基本思想和方法，具有分析、鉴赏、评价、选择、学习、设计和实现一个高级程序设计语言的基本能力；加强学生对系统软件的结构的理解，为学生毕业后从事本专业范围内的各项工作奠定坚实的理论基础。

本课程的主要内容

1. 程序设计语言

- 介绍高级程序设计语言的一些具有共性的概念和属性，主要是从设计者和实现者的角度。
- 第一、二、三、四章

2. 高级语言的编译

- 介绍语言编译中的需要考虑的问题及主要技术。
- 第五、六、七、八、九、十、十一、十二、十三章

编译程序是一个程序，该程序将用某个语言写的程序（源程序）作为输入，并翻译成为功能上等价的另一个语言程序（目标程序）

- 源语言通常是高级语言，如C、Fortran等
- 目标语言通常是低级语言，如汇编或机器语言
- 随着翻译的进行，编译程序也报告错误、警告信息以帮助程序员改错，直到翻译通过为止

英译与编译的比较

英译

1、识别出句子中的单字

2、语法分析

3、初步翻译句子的含义

4、译文修饰

5、写出最后译文

编译

1、词法分析

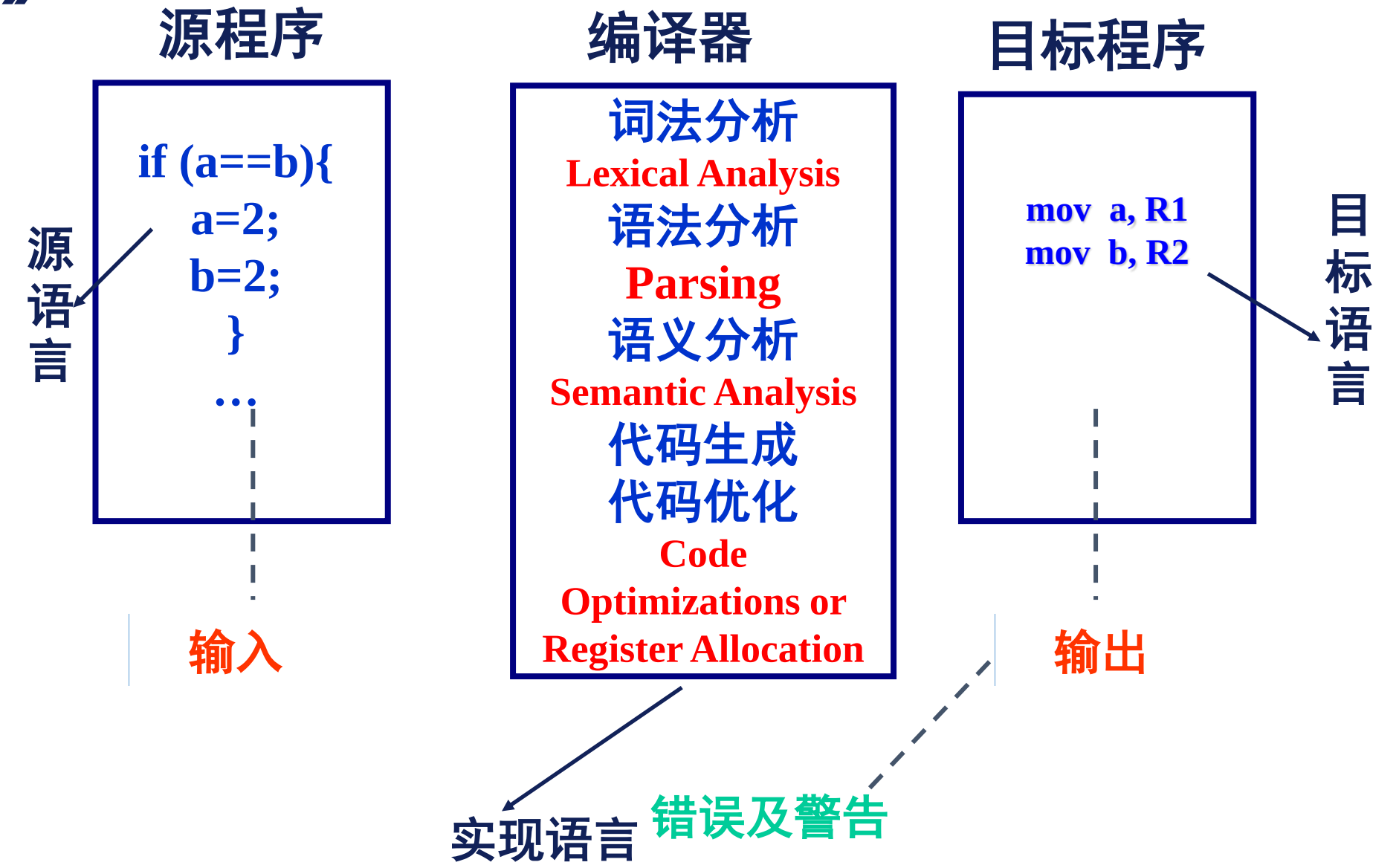
2、分析句子的语法结构

3、语义分析中间代码生成

4、优化

5、目标代码生成

编译过程


源程序
编译器
目标程序
if (a==b){
a=2;
b=2;
}
…
词法分析
Lexical Analysis
语法分析
Parsing
语义分析
Semantic Analysis
代码生成
代码优化
Code
Optimizations or
Register Allocation
mov a, R1
mov b, R2
源语言
目标语言
输入
输出
实现语言
错误及警告

Intrinsic merit

➢**Compiler construction poses challenging and interesting problems:**

- **Compilers must do a lot but also run fast**
- **Compilers have primary responsibility for run-time performance**
- **Computer architects perpetually create new challenges for the compiler by building more complex machines**
- **Compilers must hide that complexity from the programmer**
- **Success requires mastery of complex interactions**

绪论

介绍各种计算机语言的特点，高级语言的分类及主要特征和它们的发展简史。

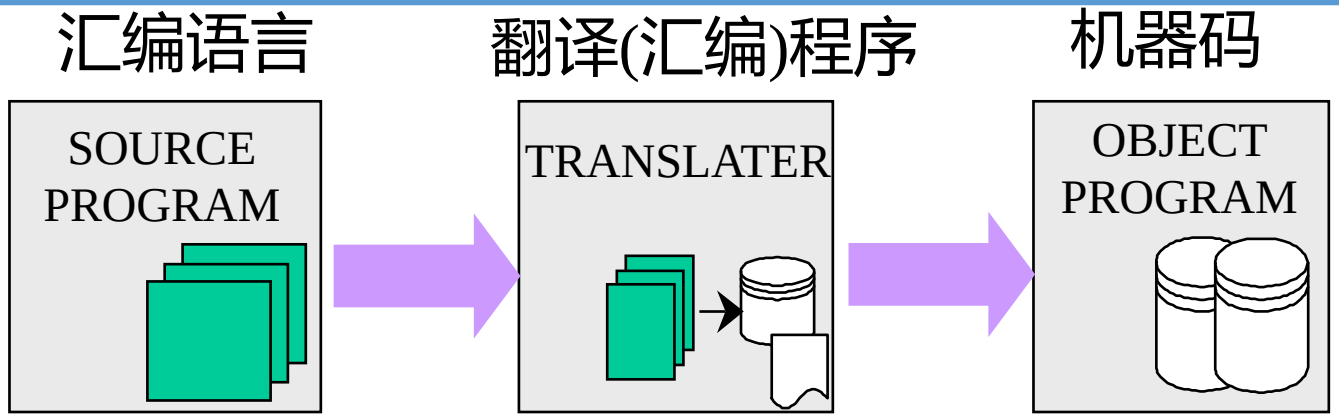
1.1引言

1. 程序设计语言的产生
   - 人机通信、人机交流
2. 程序设计语言的发展

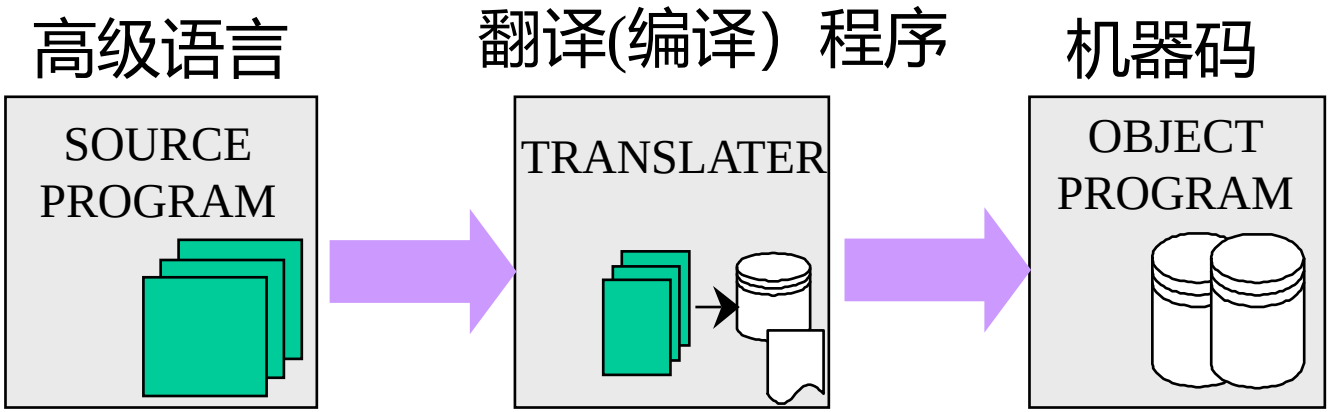
机器语言 → 汇编语言 → 高级语言

汇编与编译

- 翻译汇编语言的程序称为**汇编程序（器）**



- 翻译高级语言的程序称为**编译程序（器）**

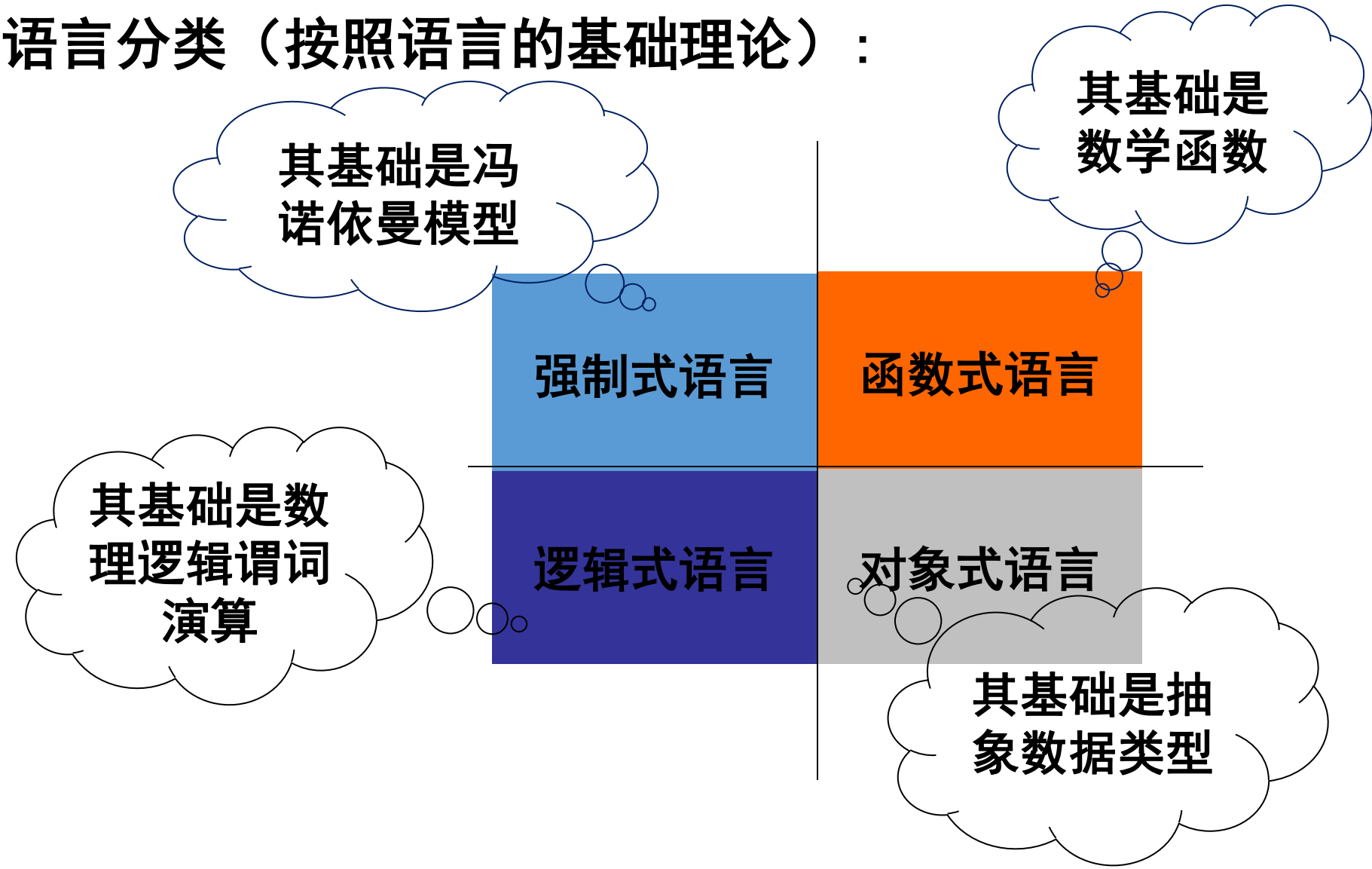


一组对比

- What is a compiler?
  - A program that translates an *executable* program in one language into an *executable* program in another language
- What is an interpreter?
  - A program that reads an *executable* program and produces the results of executing that program
- What is a language?
  - A formal notation whose syntax is described by a grammar (BNF)
  - Describes the set of “sentences” belonging to the language
  - Parsing determined sentence acceptance…. More to come…

1.2 强制式语言

语言分类（按照语言的基础理论）：



一种分类

过程式语言

FORTRAN Pascal Ada C

面向驱动，面向语句，由系列的语句组成

函数式语言

LISP ML ASL

注重程序表示的功能，而不是一个语句接一个语句的执行

从已有的函数出发构造更复杂的函数

逻辑式语言

PROLOG

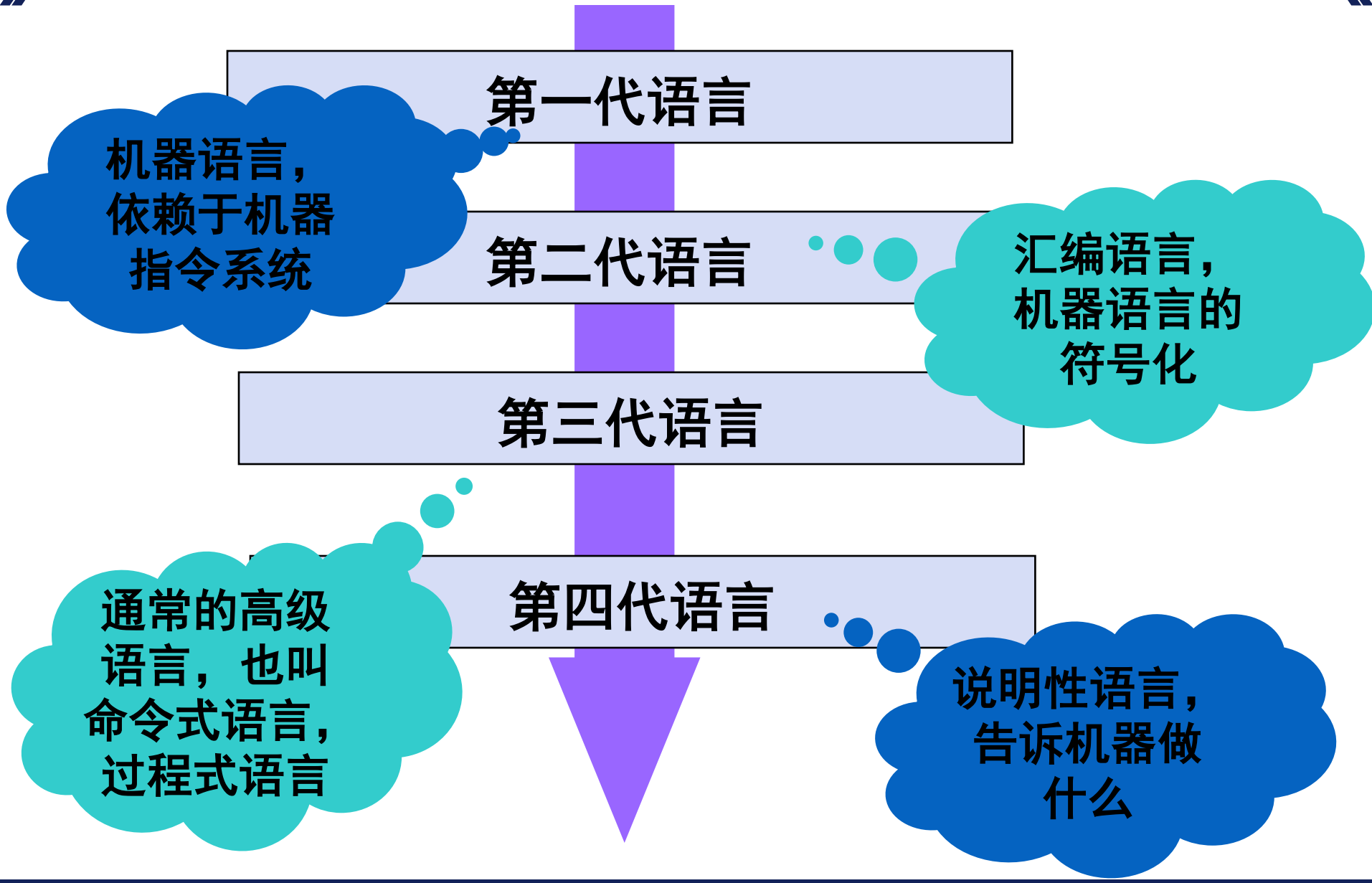
检查一定的条件，当满足时，则执行适当的动作

对象式语言

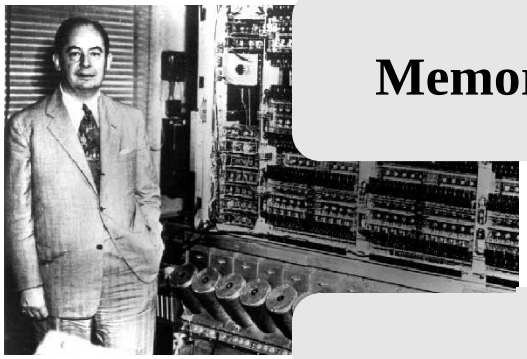
SMALLTALK C++ JAVA

封装性，继承性，多态性

按照语言的发展进程


第一代语言
机器语言，依赖于机器指令系统
第二代语言
汇编语言，机器语言的符号化
第三代语言
第四代语言
通常的高级语言，也叫命令式语言，过程式语言
说明性语言，告诉机器做什么

冯.诺依曼体系结构



Memory Data Register

Monitor
Printer
LED
Disk

Small, temporary storage

Memory Address Register

Instruction Register

Keyboard
Mouse
Scanner
Disk

Program Counter

Arithmetic and Logic Unit

PC

IR

冯.诺依曼John von Neumann

1903-1957, 原籍匈牙利，22岁获数学博士学位。先后执教于柏林大学和汉堡大学，1930年赴美国，历任普林斯顿大学、普林斯顿高级研究所教授，美国原子能委员会委员。20世纪最重要的数学家之一，被称为“计算机之父”和“博弈论之父”。

若人们不相信数学简单，只

因他们未意识到生命之复杂

1945年6月，冯. 诺伊曼与戈德斯坦、勃克斯等人，联名发表了一篇长达101页纸的报告，即计算机史上著名的“101页报告”，这是现代计算机科学发展里程碑式的文献。明确规定用二进制替代十进制运算，并将计算机分成五大组件。

特点及表现

冯.诺依曼体系结构的特点

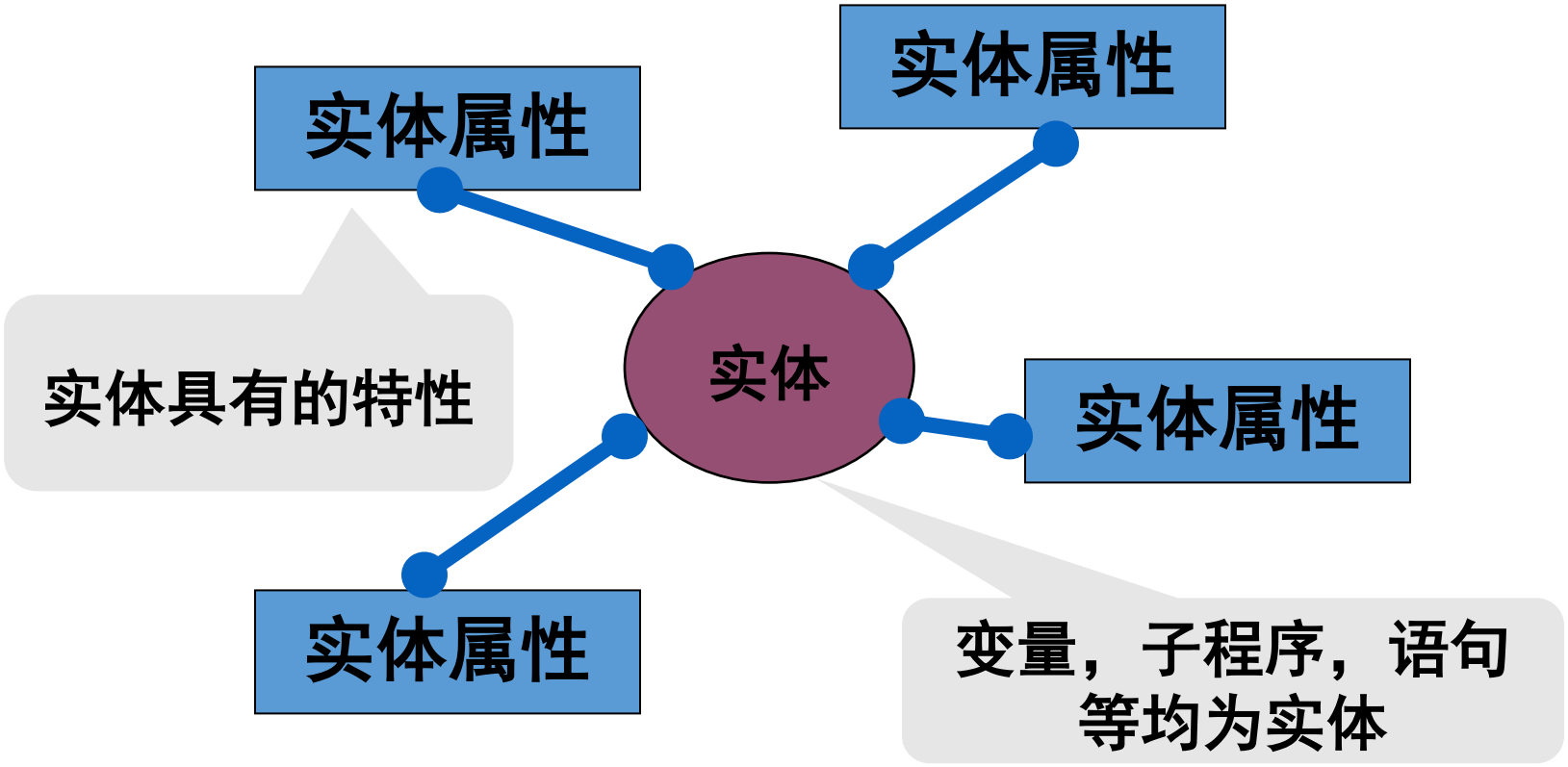
- 数据或指令以二进制形式存储
- “存储程序”的工作方式
- 程序顺序执行
- 存储器的内容可以被修改

冯.诺依曼体系结构在命令式语言上的表现

- 变量 存储单元及它的名称由变量的概念来代替，可以代表一个或一组单元，可以修改
- 赋值 计算结果必须存储
- 重复 因语句顺序执行，指令存储在有限的存储器中，完成复杂计算时必须重复执行某些指令序列

绑定（Binding)概念

一个实体（或对象）与其某种属性建立起某种联系的过程，称为绑定



关于绑定（Binding)的一些概念

- 描述符：用以描述实体的属性的符号、语句或表格等。亦即实体到属性的映象
- 凡是在编译时能确定的特性，称为静态特性；若绑定在编译时完成，运行时不改变，称为**静态绑定**
- 凡是在运行时才能确定的属性称为动态的。若绑定在运行时完成，称为**动态绑定**
- 例：动态数组和静态数组。数组的属性有保留其值的存储区

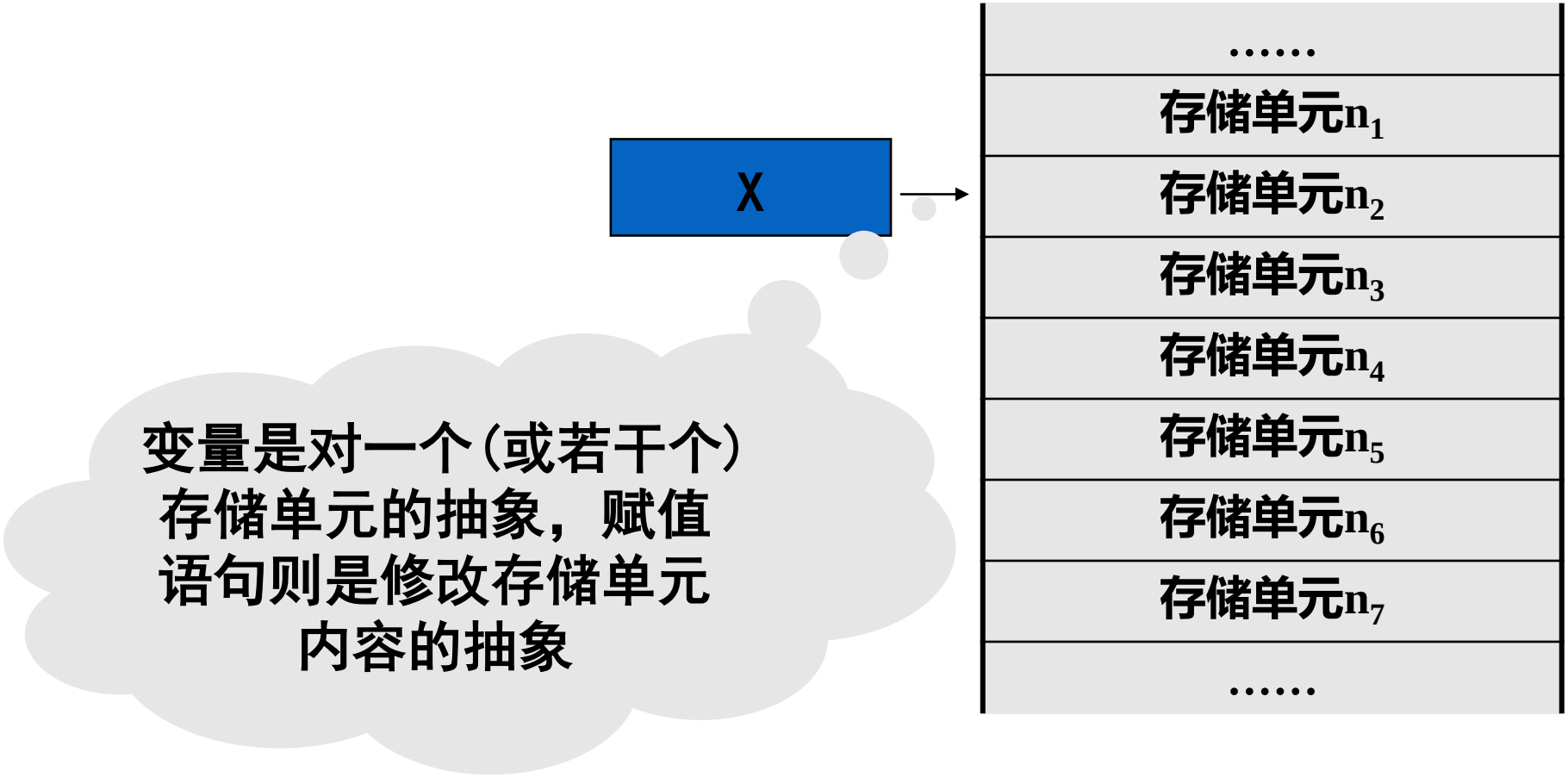
FORTRAN的integer

Pascal中的integer类型

可改变数组大小或释放数组所占空间，如用变量或表达式指定上下界

数组的维数和大小在建立到运行结束的过程中不再改变

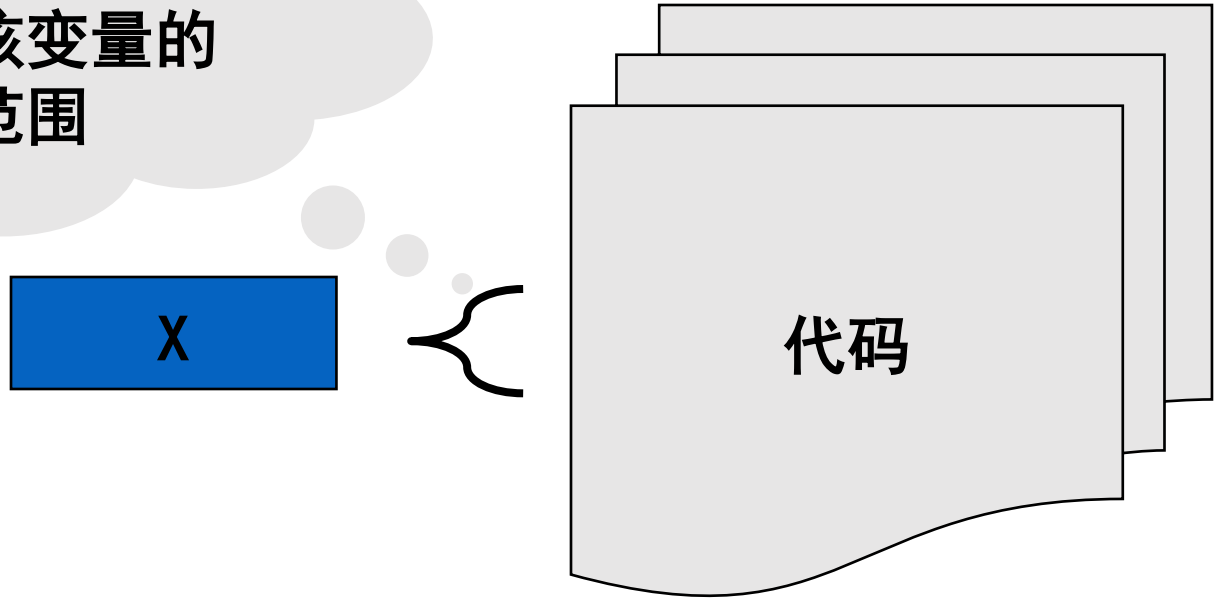
变量(Variable)



- 变量除名字外，具有四个属性：作用域（Scope）、生存期(Lifetime)、值(Value)和类型(Type)

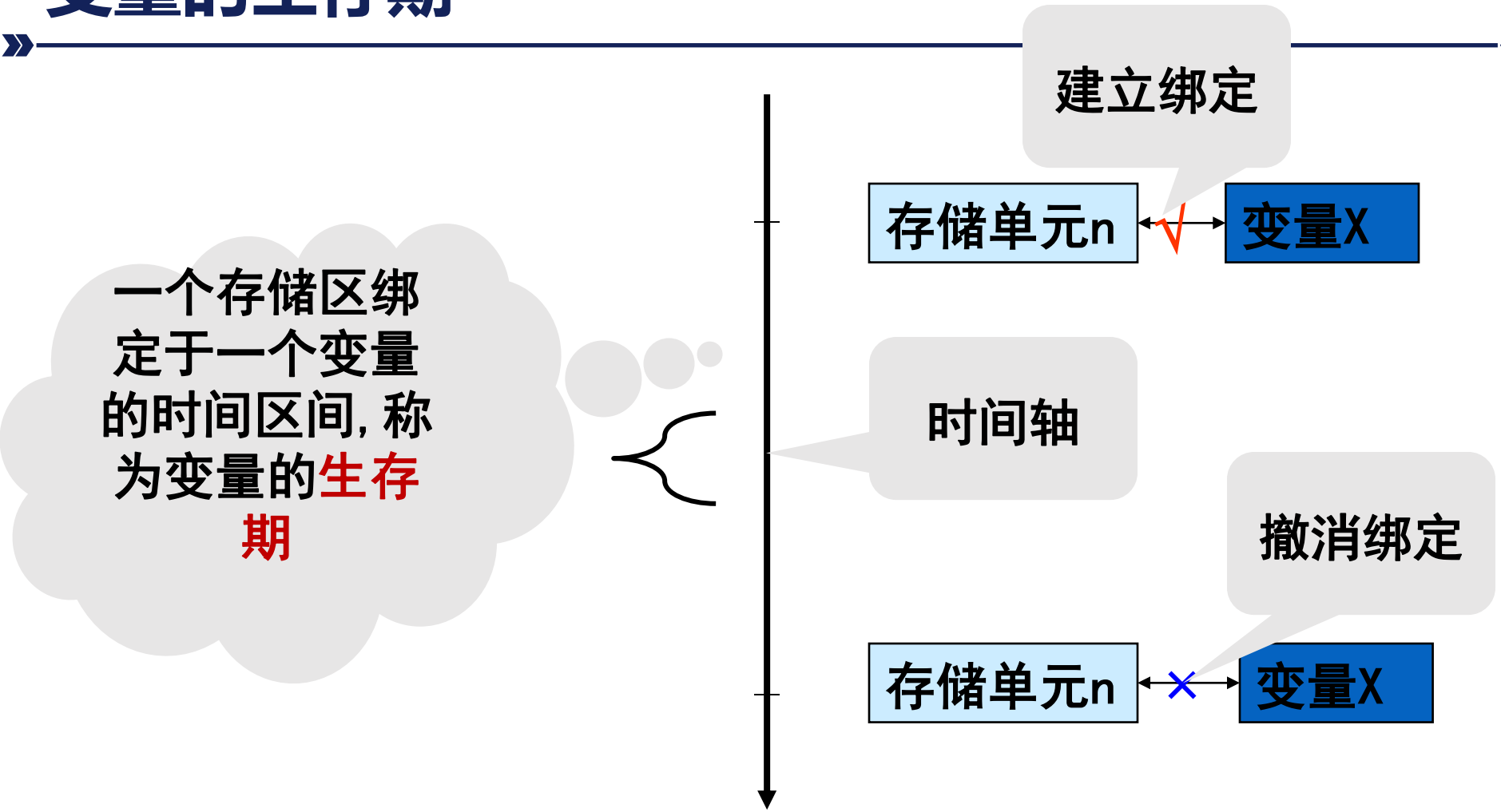
变量的作用域

变量的**作用域**是指可以访问该变量的程序范围



- **静态作用域绑定**：按照程序的语法结构定义变量的作用域
- **动态作用域绑定**：按照程序的执行动态地定义变量的作用域

变量的生存期



数据对象(Object)：存储区和它保存的值

分配(Allocation)：变量获得存储区的活动

变量的值

定义

即变量对应存储区单元的内容

主要问题

关于变量的值的主要问题：

- 匿名变量的访问通过指针实现
- 变量与它的值的绑定是动态的
- 符号常数的值不能修改（冻结）
- 变量的初始化，几种处理方法：
  - ✓不初始化则出错
  - ✓随机
  - ✓缺省值0

赋值操作可修改

constant pi=3.1416

“先赋初值再引用”原则

变量的类型的绑定

静态绑定：通过说明语句完成

如：Pascal、Fortran、C

动态绑定：执行时隐式说明，且动态变化

如：APL

可读性差

```
A←5                    //整型
    → A                //标号、转到A
A←1 2 51 0   // 一维数组
```

动态检查：

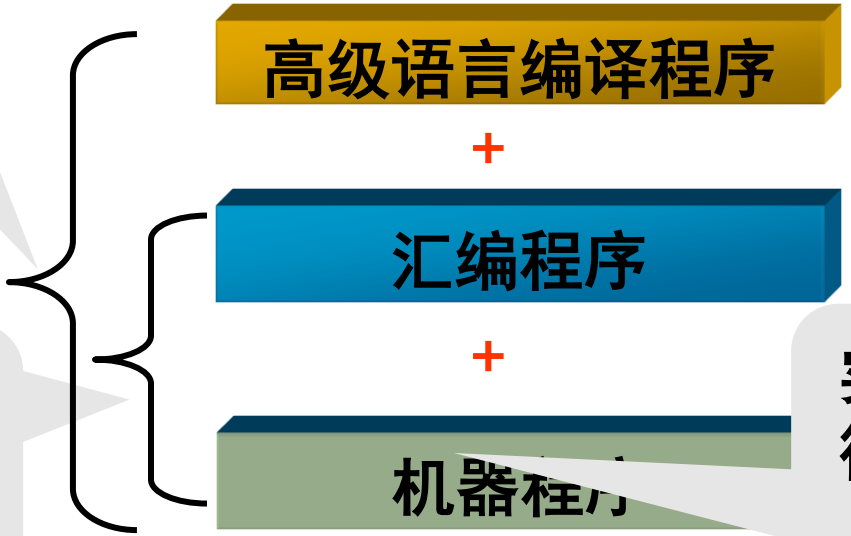
```
A[2:3]←5      //二维数组（执行错误）
A←0           //动态检查
A←B+C         //
```

虚拟机

定义：虚拟机是由软件实现的机器

执行高级语言的虚拟机

执行汇编语言的虚拟机

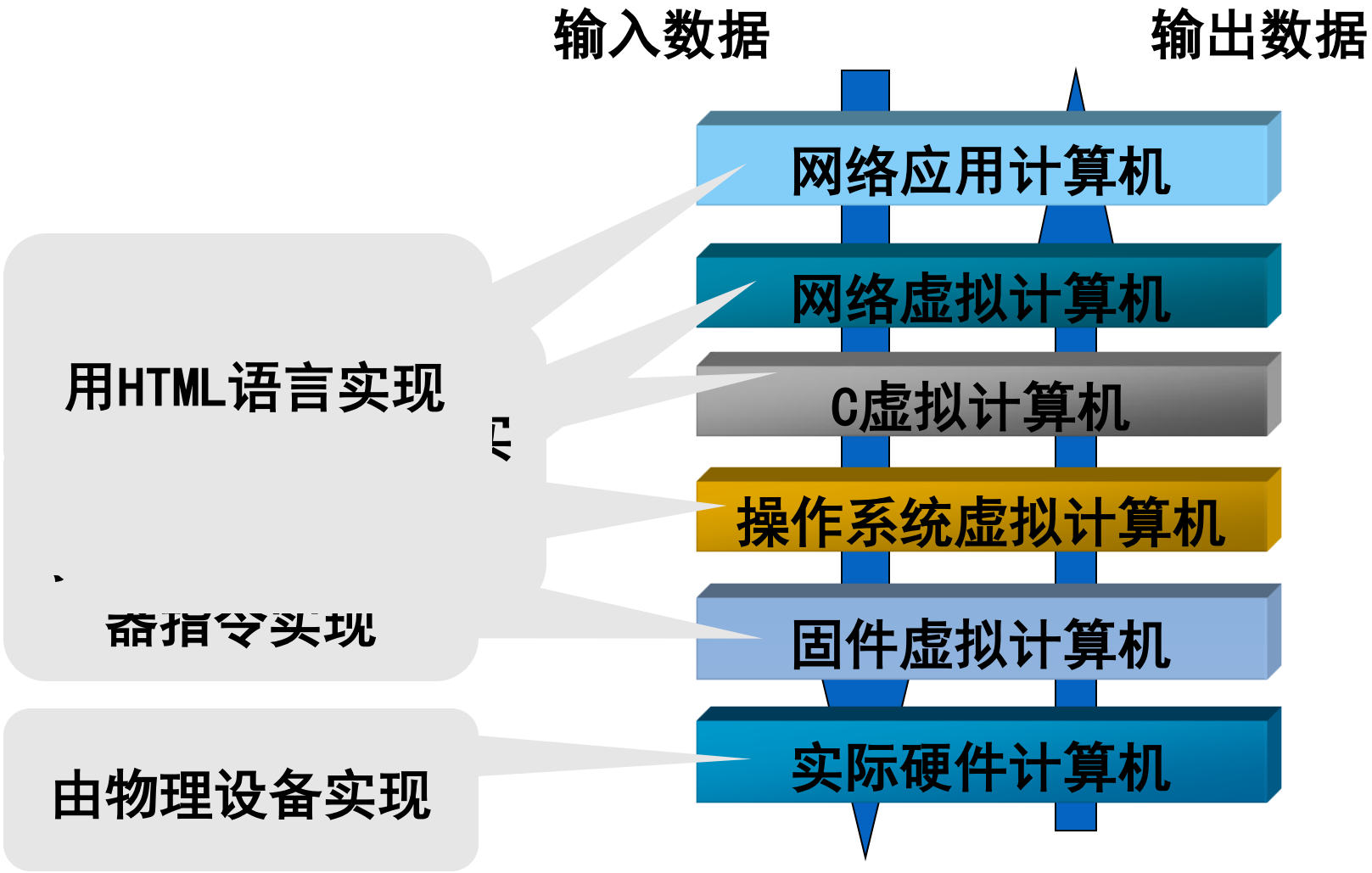


实际机器，执行二进制机器语言

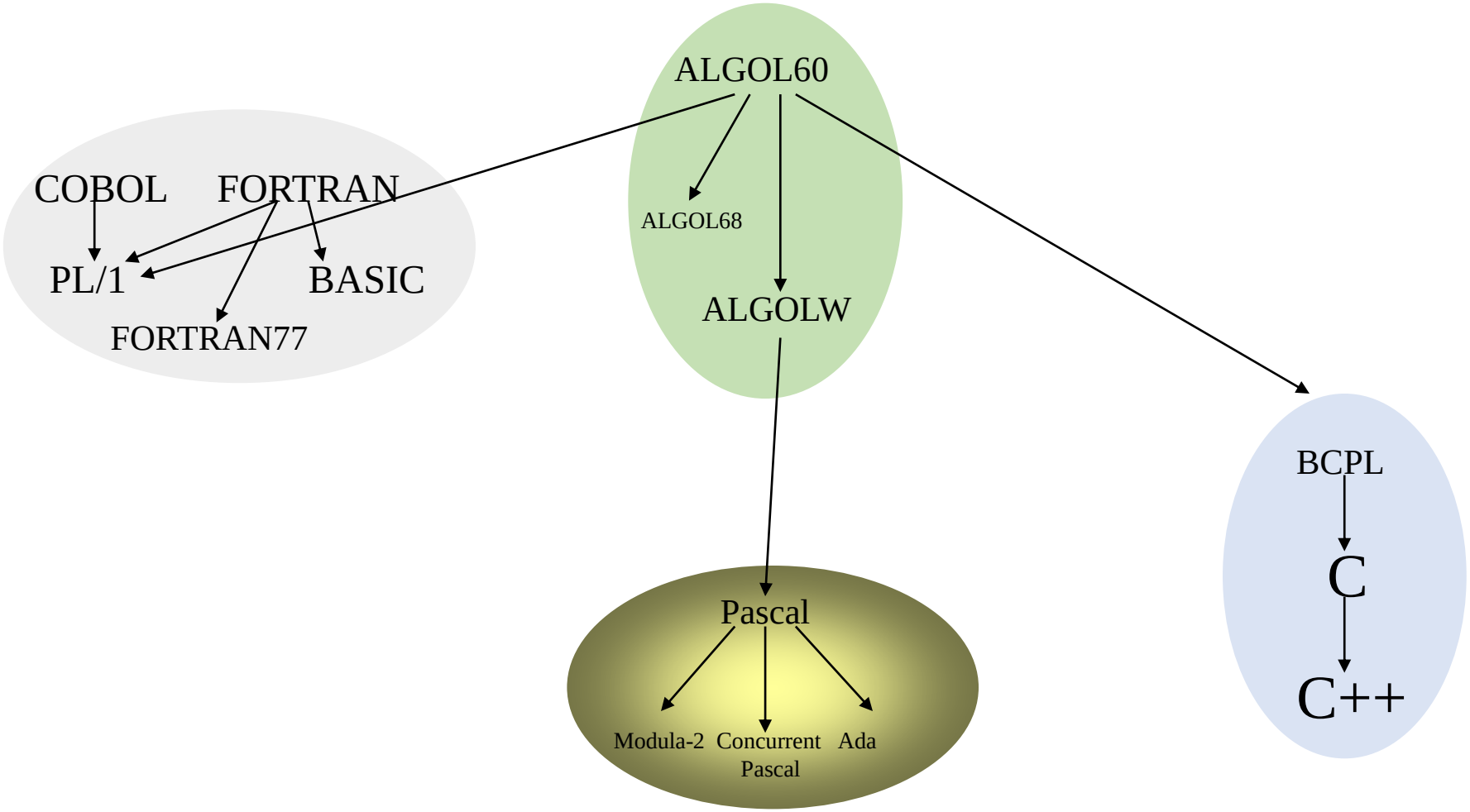
虚拟机与抽象机的差别

虚拟机是在一台实际机器上配置某种软件扩大其功能而实现的；
而抽象机仅仅是一个抽象模型，并不要求与之匹配的实际机器存在

一个网络应用程序的虚拟机层次



主要的强制式语言及其关系



1.3 程序单元

与程序单元有关的概念

- 程序单元(Unit):程序执行过程中的独立调用单元；如子程序、分程序、过程等。
- 单元的表示(Unit Representation)
  - ✓ 在编译时，单元表示是该单元的源程序。
  - ✓ 运行时，单元表示由一个代码段和一个活动记录组成，称为单元实例。
- 活动记录(Activation Record):执行单元所需要的信息,以及该单元的局部变量所绑定的数据对象的存储区。

1.3 程序单元

与程序单元有关的概念

- 非局部变量（Nonlocal Variable）:一个程序单元可以引用未被本单元说明而被其它单元说明的变量。
- 引用环境:局部变量+非局部变量。
- 别名(Alias):同一单元的引用环境中有两个变量绑定于同一数据对象，称这些变量具有别名。
- 副作用(Side Effect):对绑定于一个非局部变量的对象进行修改时，将产生副作用。
- 程序单元可以递归激活,从而一个单元可以有很多个实例,但代码段相同。不同的仅仅是活动记录。

1.4程序设计语言发展简介

随着计算机技术的发展，计算机应用也日益广泛，已经渗透到社会的各个领域，对程序设计语言也提出了新的要求（诸如可维护性，可靠性，可移植性等），从而促进了语言的发展。

1. 早期的高级语言（50年代）

原因：计算机资源稀少而昂贵！

FORTRAN语言

FOR = **FOR**mula的前三个字母
TRAN = **TRAN**slation的前四个字母

主要特征：

- 主要用于科学计算
- 子程序独立编译
- COMMON语句实现了模块之间的通信

ALGOL 60语言

ALGO = **ALGO**rithmic的前四个字母
L = **L**anguage 的第一个字母
60 = 宣布的时间

主要特征：

- 主要用于科学计算
- 引入了分程序结构和递归过程
- 采用巴科斯-诺尔范式（BNF）形式描述语法

COBOL语言

CO = COmmon的头二个字母
BO = Business-Oriented 的第一个字母
L = Language的第一个字母

主要特征：

- **广泛应用于各种事务处理领域**
- **引入了文件和数据描述**
- **类自然语言程序描述**

2.早期的突破

典型语言

LISP语言

- 函数式语言
- 具有很强的符号处理能力
- 统一的数据结构（表）
- 数据和程序统一的表示方法
- 其基础是函数和函数作用

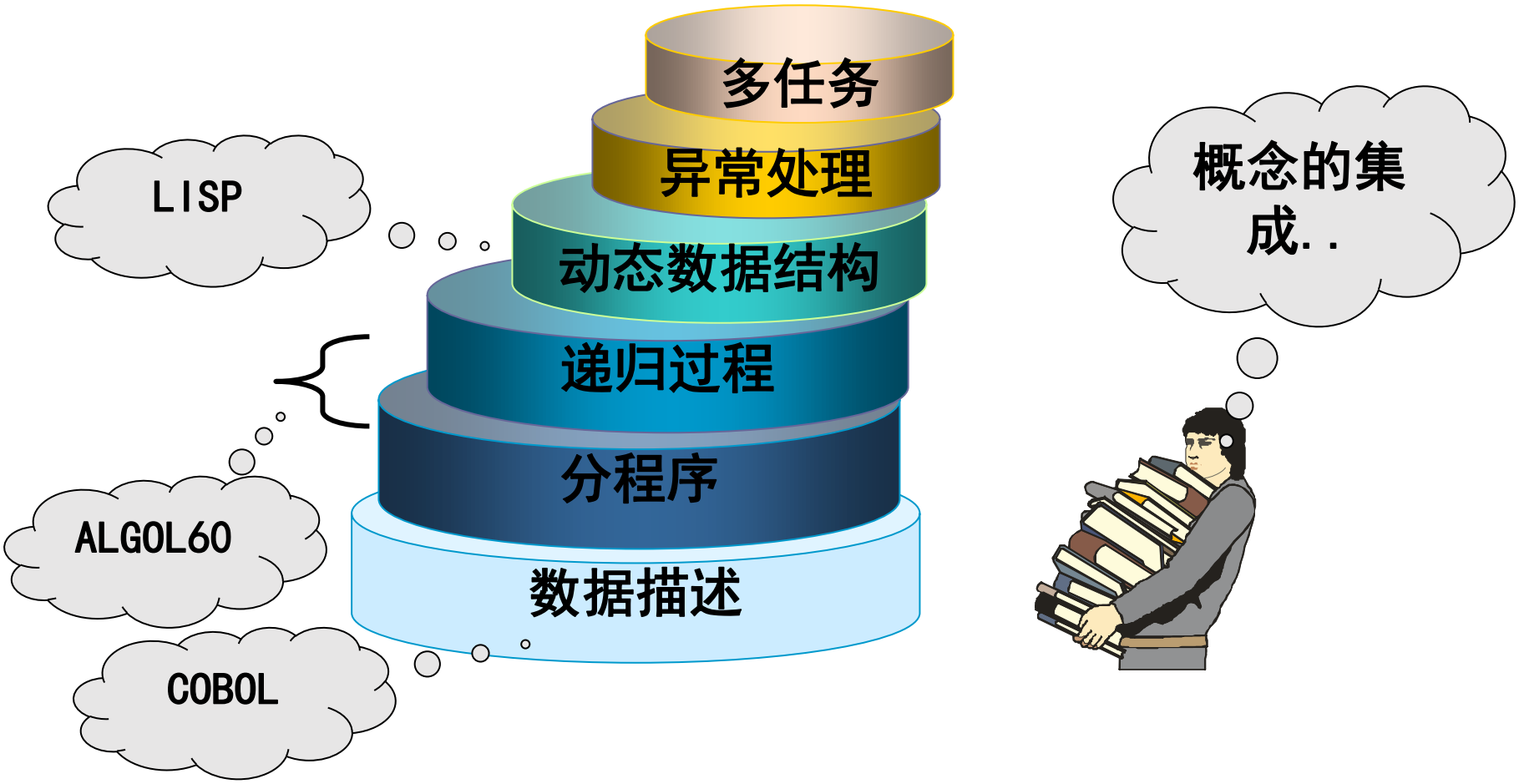
APL语言

- 支持函数式程序设计风格
- 广泛应用于涉及大量矩阵运算的科学计算中
- 具有丰富的操作符

SNOBOL语言

- 主要用于字符串处理
- 给出了一种与机器无关的宏功能，增加了程序的可移植性
- 成功应于文本处理

3.概念的集成（64年）

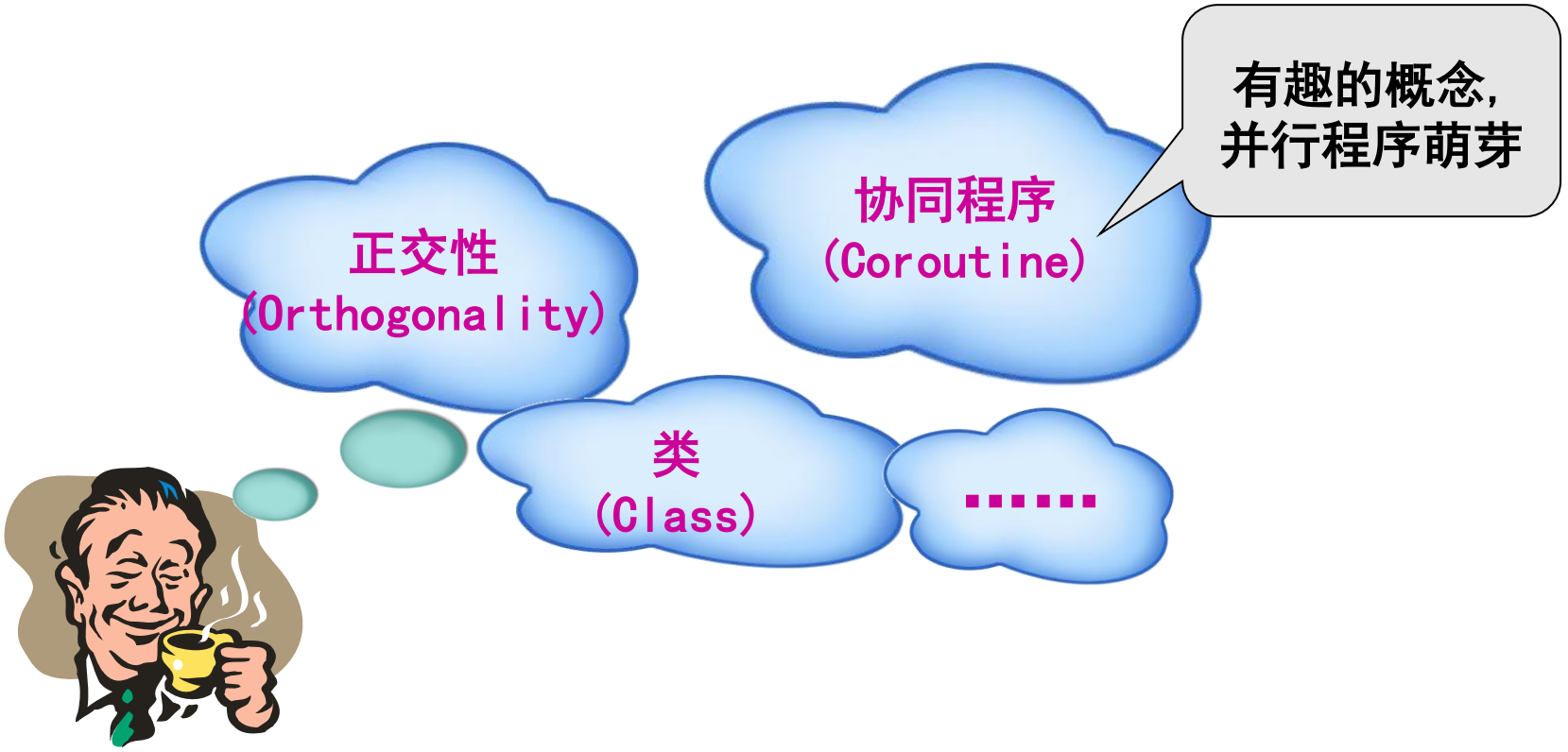


PL/1语言

主要特征：

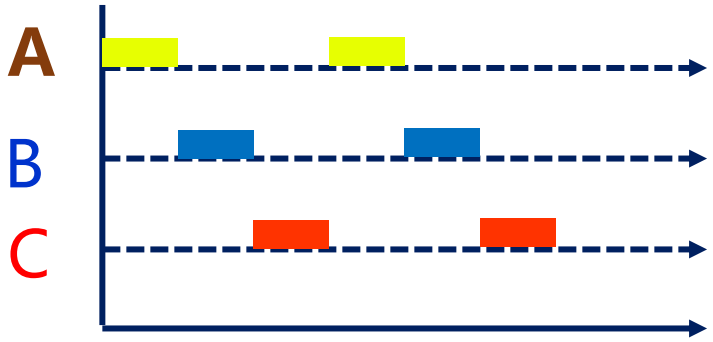
- 所有语言概念之大全
- 分程序概念和递归过程
- 数据描述机能
- 动态数据结构
- 异常处理
- 多任务机能
- 可用于科学数值计算，数据处理和开发系统软件
- 没有（难以）得到广泛的应用

4.再一次突破(60年代后期)



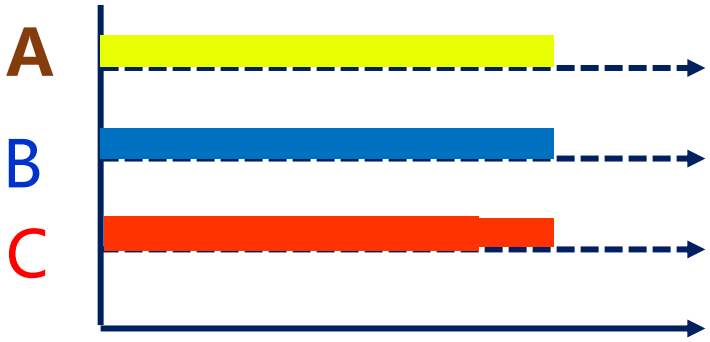
并发和并行

并发(concurrency)



- 单处理器
- 两个或多个事件在同一时间间隔发生
- 同一实体上的多个事件

并行(parallelism)



- 多处理器
- 两个或者多个事件在同一时刻发生
- 不同实体上的多个事件

典型语言

ALGOL 68语言

- 以零型文法描述
- 引入正交性和通用性原则

SIMULA67 68语言

- 应用于模拟领域
- 增加了一个特殊结构—协同程序
- 引入了类的概念

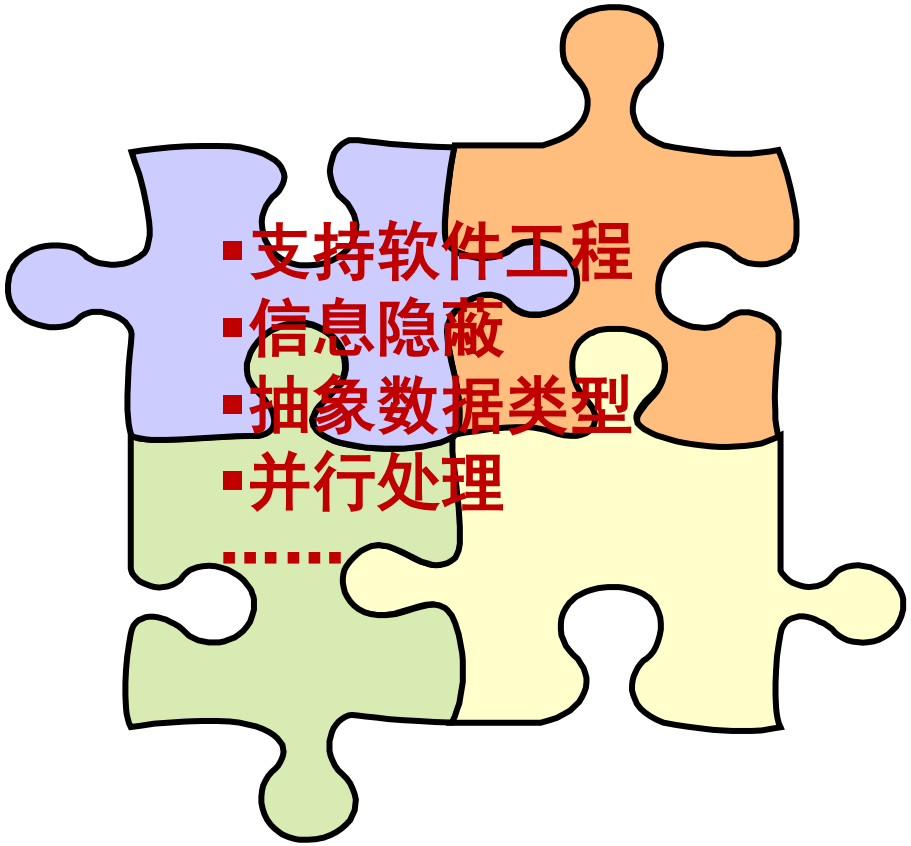
PASCAL语言

- 具有明显的简洁性
- 体现结构程序设计思想
- 具有用户自定义类型

BASIC 语言

- 简单易学
- 交互式工作环境
- 解释执行

5.大量的探索



典型语言

MODULA-2 语言

- 支持模块结构，模块可以独立编译
- 面向实时系统和并行系统综合功能

C 语言

- CPL→BCPL→B→C
- 具有高级语言和低级语言的优点
- 应用于各种领域

6.Ada语言

主要特征：

- 面向专门领域的特殊要求
- 在PASCAL基础上引入了一个不大的，容易理解的概念集合
- 直接体现了许多现代软件设计方法学观念
- 提高了程序的可读性，可靠性，可维护性

7.第四代语言

主要特征：

- 面向问题
- 表达力更强，使用更方便，更接近于问题的描述
- 着重关心的是”做什么”

例如：数据库查询语言SQL

8.网络时代的语言

JAVA语言

- 面向因特网
- 保留了C++语言的语法、类和继承等基本概念
- 其语法和语义比C++语言更合理

C#语言

- 面向对象的编程语言
- 每个对象自动生成为一个COM对象，C#可以调用现有的无论由什么语言编写的COM对象
- 高效率、网络开发

Python

特点

- 解释型语言，解释器易于扩展，可以使用C或C++扩展新的功能和数据类型
- 提供高级数据结构，简单有效地面向对象编程
- 可用于可定制化软件中的扩展程序语言，丰富的标准库
- 提供了适用于各个主要系统平台的源码或机器码

9.新一代程序设计语言

主要特征：

- 以抛弃冯. 诺依曼模型为出发点
- 包括函数式, 对象式, 逻辑式程序设计语言

中文编程语言

易语言
- 一门计算机编程语言，以"易"著称，以中文作为程序代码表达的语言形式。早期版本的名字为E语言。

习语言
- 习语言即中文版的C语言

A语言
- 中文版的pascal语言，是一个高级解释性编程语言

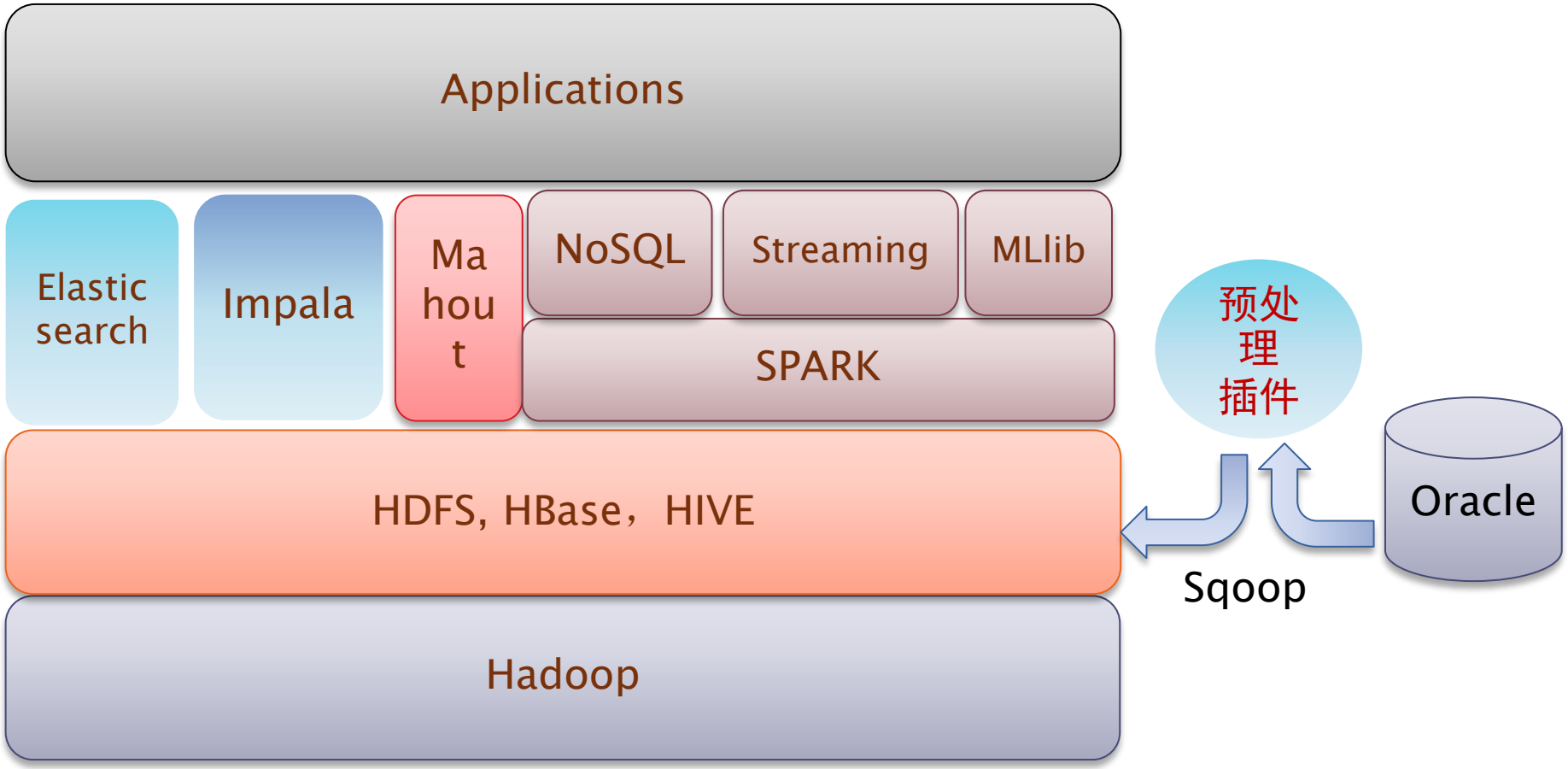
雅奇MIS
- 无代码编程的领先者

创新LOGO
- CX-LOGO语言独创的流程图工作方式和过程库的建立，可方便的使用"搭积木"的方法，构建"知识"，使学习更容易、操作更简便

O汇编语言
- 是一门汇编语言，它具有传统汇编语言的基本特点

搭建之星
- 原名"发烧积木"属于搭建式的编程工具，完全可视化编程

大数据平台架构


Applications
Elastic search
Impala
Ma hou t
NoSQL
Streaming
MLlib
SPARK
HDFS, HBase，HIVE
Hadoop
预处理插件
Oracle
Sqoop

习题

1. 必做题:

   1-2、1-6、1-11

2. 思考题:

   1-3、1-5、1-10