FSE598 前沿计算技术

模块 2　数据与数据处理
单元 1　编程语言基础
第 1 讲　编程范式

讲座的英文版内容基于本书：
Y. Chen 《编程语言入门：C、C++、Scheme、Prolog、C# 和 Python 编程》(Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and Python)，第 6 版，Kendall Hunt Publishing Company，2019 年。
https://www.public.asu.edu/~ychen10/book/IntroPl.html

本讲大纲

学习

- 不同的计算范式
- 语言设计与性能
- 编程语言与范式的发展
- 计算范式的特点

编程语言的各个方面

我们使用什么标准（指标）来评估编程语言？

1. 可靠性
2. 可读性
3. 可写性
4. 可复用性
5. 效率

- 重要性取决于你使用语言的场合。
- 典型的选择是：重要性从高到低

但是，如果你正在开发一个实时系统并且必须赶上最后期限，你可能就不得不将效率作为最重要的因素。

编程语言的各个方面

一种语言的哪些特性可以改进用这种语言编写的程序的哪些性能指标？

性能指标 / 语言特性	效率	可读性/可重用性	可写性	可靠性
简单性/正交性	⊕	⊕	⊕	⊕
良好的控制结构	⊕	⊕	⊕	⊕
良好的数据类型和数据结构	⊕	⊕	⊕	⊕
良好的语法设计		⊕	⊕	⊕
支持抽象		⊕	⊕	⊕
可表达性			⊕	⊕
强类型检查			⊖	⊕
异常处理				⊕
受限（限制）别名/指针	⊖	⊕	⊖	⊕

其中⊕表示正面影响，⊖表示负面影响。

编程语言的发展

受以下各类因素的影响

- 硬件的发展——早期强调执行效率；
- 编译器技术的发展；
- 编写大型程序的需求和能力：
  - 代码重用；
  - 组件重用；
  - 系统重用
  - 促进可重用代码的发现
  - 重用代码的可信度
  - 网络共享
  - ……

编程语言的发展（续）

存储程序概念（冯诺依曼机）

这一概念是在 20 世纪 40 年代后期发展起来的。目前仍然是最流行的计算模型。

- **程序**是存储在内存中的一系列指令。
- CPU 以指定的顺序解释程序。
- 在最底层，程序是位序列（二进制数）的**机器码**。数据也通过二进制方式存储。

下一步：使用助记符（某些符号），让编写更复杂的程序（**汇编语言**，例如 Java 的字节码、Intel 汇编和 MIPS）：

1. 简单变量
2. 条件汇编
3. 宏，相对寻址

需要程序（汇编程序）转换为机器代码才能执行。 仍然非常低级。

编程语言的发展（续）

随着任务规模越来越大，机器功能越来越强大，
就需要开发更高级别的编程语言。
示例：Autocode、FORTRAN、COBOL、LISP、Scheme、C、…
Manchester Mark I 上的 Autocode，1952 年：

- 第一款高级编程语言
- 单字母标识符(26)
- 简单公式：a+b, c-d 2*a+3*b*c-d

FORTRAN：IBM Mathematical FORmula TRANslating 系统

- 不同类型的变量（实数、整数、数组）；
- 过程
- 条件（if-then、no if-then-else）
- 简单控制结构

但仍然非常低级；对机器的依赖程度很高。
有一些难以理解的特征。
其中包括面条式编程，或单体式编程。

编程语言的发展（续）

结构化编程的出现：Algol、Pascal、C

结构化编程的主要理念是

- 更抽象。
- 关注程序中的结构——使用过程将程序分成大小合理的部分，而不仅仅是一种代码重用方式。
- 鼓励使用不同的范围（即局部变量）。
- 为过程提供更好的支持
- 更优秀的数据结构
- 更好的控制和循环结构。 认为“Goto”是有害的。 *“程序员的素质与他们程序中 go-to 语句的使用密度成反比。”* ——Dijkstra。

面向对象程序设计

- 抽象/封装
- 代码重用/继承
- Simula、Smalltalk、Eiffel、C++、C#、Modula-3、Python、Java 等

编程语言的发展（续）

并行和分布计算与编程

- 并行/多线程编程：并行运行的多个任务或线程。Python、Java、C#、Concurrent C 和 C++， 当然，VIPLE!
- 多个 CPU 处理相同的任务（共享和分布式内存）：CSP、occam、Linda、Emerald、Ravan、Core。

面向服务的编程，基于 OO、+ Web 技术

- XML/XML Schema
- RDF/RDFS
- OWL
- WSDL
- UDDI
- SOAP
- BPEL
- Xlang
- WS-CDL

VIPLE 是一种支持下列范式的编程语言

- 通用控制流编程（命令式）
- 面向服务的计算，支持 RESTful 和 WSDL 服务
- 并行/多线程编程，确保底层线程安全性
- 事件驱动编程，具有内置和自定义事件
- 工作流和可视化编程
- 物联网和机器人编程

范式

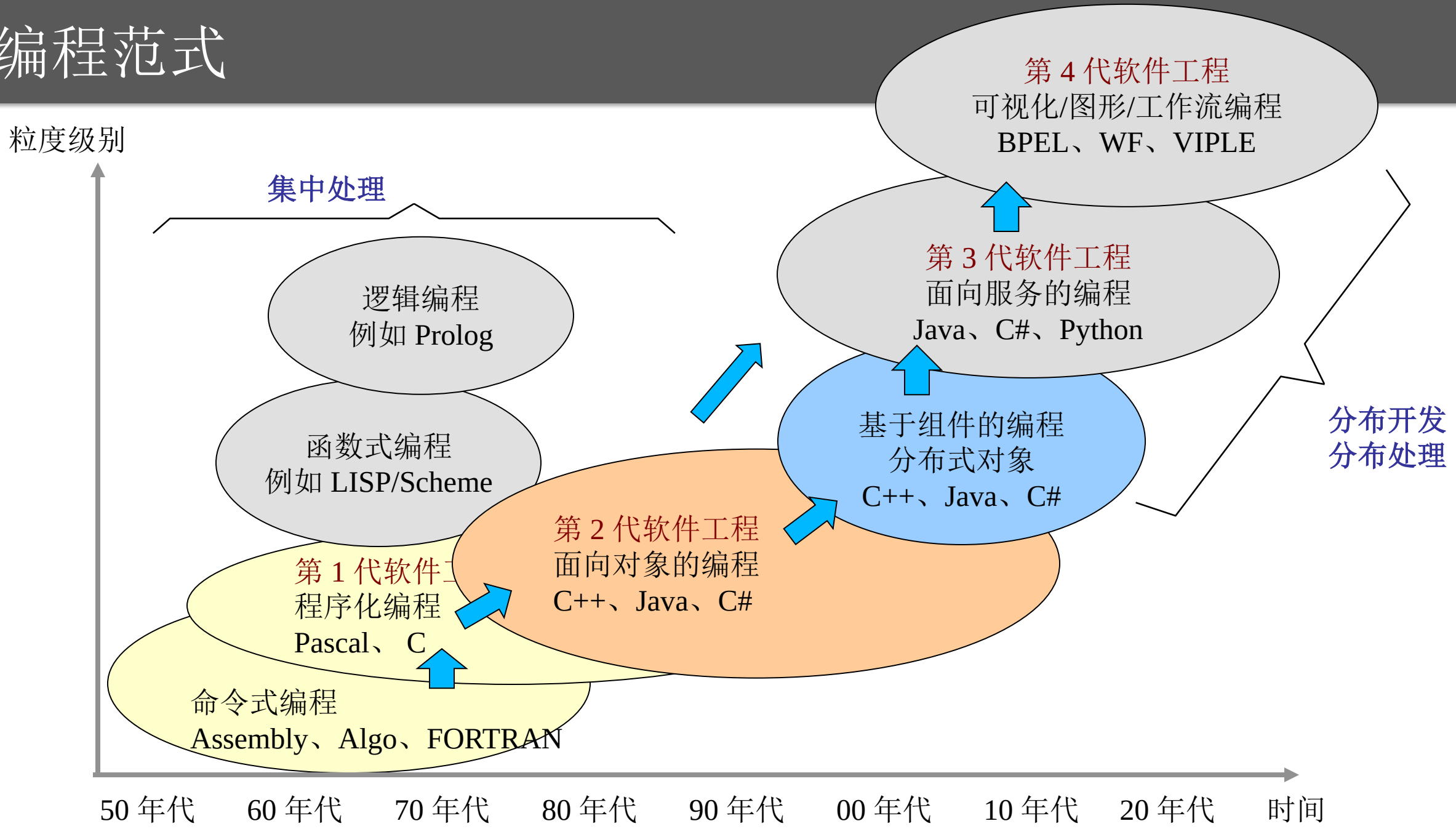
在编程语言定义中：**范式**是指如何表达计算或算法的基本原则。

1. 命令式/程序式，例如，Assembly、Fortran、Ada、Pascal、C、
2. 面向对象：Smalltalk、Java、C++、Python、C#
3. 函数式/应用式：Scheme/LISP、ML、LINQ
4. 逻辑/声明式编程：Prolog，
5. 面向服务：扩展面向对象的编程语言
6. 多范式：许多现代语言都支持多种范式，例如 Python、Java、C# 和 VIPLE。

五大编程范式

- **命令式/程序式：**以一步一步的顺序方式，对命名数据进行完全说明和完全受控的操作。
- **函数式/应用式：**关注更高级的抽象（没有编程细节）、更简单的语义，更接近数学函数和引用透明性（没有副作用）。
- **逻辑/声明式：**程序是一组关于对象的事实、关于对象的规则，定义并查询对象之间的关系。目标是完全摆脱编程。
- **面向对象：**将程序的状态封装在对象中，只能通过针对这些对象定义的操作来访问这些对象。其他功能包括继承和多态。
- **面向服务：**将开发方分为三方：服务提供方、服务请求方和服务代理方。服务和应用程序通常部署到互联网上。

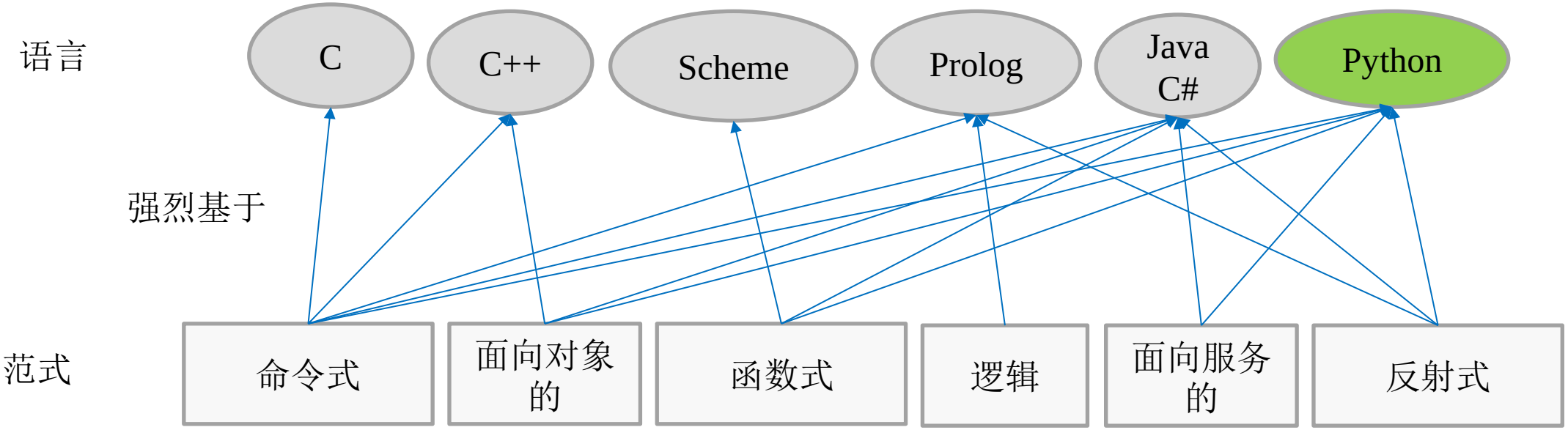
编程范式


粒度级别
集中处理
第 4 代软件工程
可视化/图形/工作流编程
BPEL、WF、VIPLE
第 3 代软件工程
面向服务的编程
Java、C#、Python
逻辑编程
例如 Prolog
函数式编程
例如 LISP/Scheme
基于组件的编程
分布式对象
C++、Java、C#
分布开发
分布处理
第 2 代软件工程
面向对象的编程
C++、Java、C#
第 1 代软件工
程序化编程
Pascal、 C
命令式编程
Assembly、Algo、FORTRAN
50 年代
60 年代
70 年代
80 年代
90 年代
00 年代
10 年代
20 年代
时间

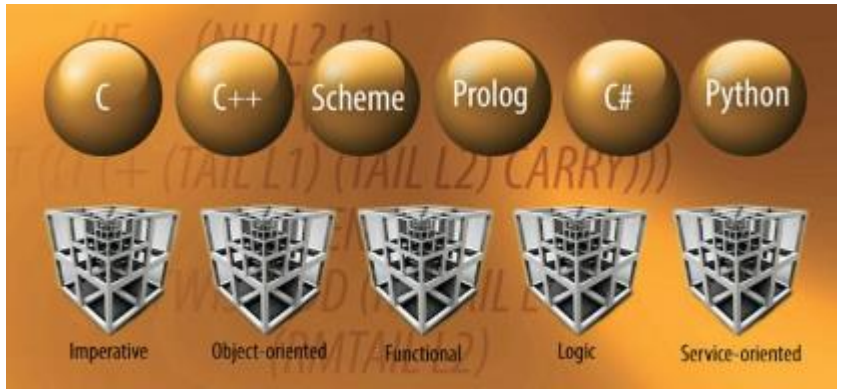
计算范式的特点


分布
对象
面向服务的
计算
多线程
编程
并行处理
同步
资源共享
面向对象的
编程
抽象数据类型
封装
继承
动态/延迟绑定
函数式
编程
无状态
无副作用
强制模块化设计
逻辑
编程
数据库
关系
查询和匹配
推理
自主服务
松散耦合
远程对象
基于组件的构成
远程调用和绑定
开发分离（分布式开发）
开放标准和协议
独立于平台
可重用服务的存储库
可通过互联网搜索的服务
自动发现和绑定
协作协商
动态重组
人工智能和基于本体的推理

编程语言及其范式



Y. Chen 《编程语言入门：C、C++、Scheme、Prolog、C# 和 Python 编程》(Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and Python)，第 6 版，Kendall Hunt Publishing Company，2019 。



反射范式是指具有检查、内省和修改自身结构和行为能力这一过程的语言。

https://www.wikiwand.com/en/Reflective_programming