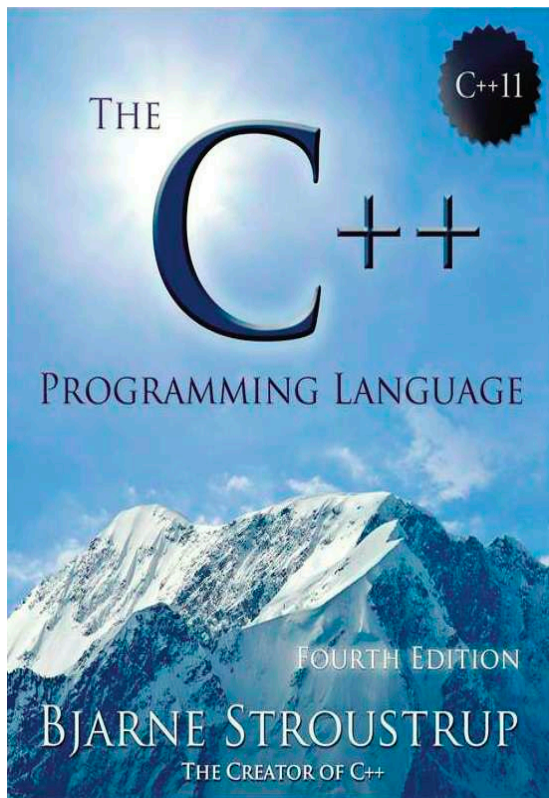




C++ 语言程序设计

潘建瑜

华东师范大学 数学科学学院



- 掌握 C++ 语言的基本语法规则
- 熟练阅读和分析 C++ 程序源代码
- 掌握类与对象的基本思想与实现方法
- 掌握**算法**的基本概念和设计方法
- 培养面向对象的程序设计**思维**和**能力**
- 掌握基本的编程**技巧**和**调试**技术

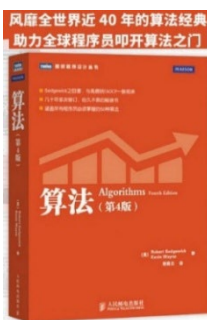
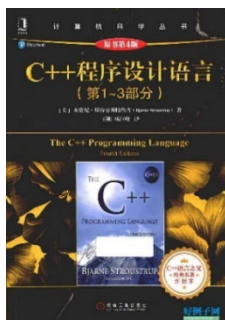
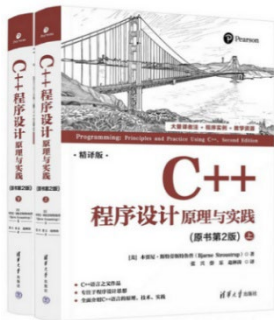
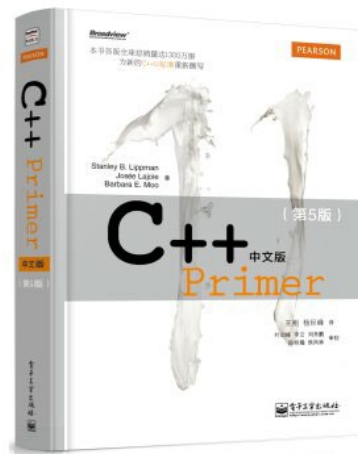
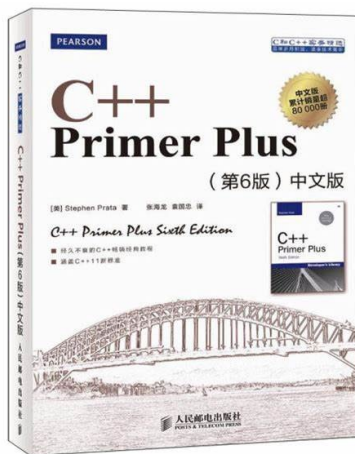
教材：课堂讲义+参考资料

《C++ 语言程序设计》简明讲义*

潘建瑜

华东师范大学·数学科学学院

2026 年 9 月



>>> AI 辅助

>>> <https://cppreference.cn> (参考手册, 含各种标准库)

课程基本信息

□ 上机时间：周一晚上 11、12

地点：数学楼 200B（二楼最东边）

□ 答疑时间：周一 14:40 — 15:40 + 上机时间

地点：数学楼 335 / 200B

□ 课程主页：<https://math.ecnu.edu.cn/jypan/Teaching/Cpp>

□ 成绩评定：平时成绩 60% + 期末考试 40%

平时成绩：**上机测验**（1大+2小）+ 上机作业 + 课堂表现



程序设计语言介绍

- 程序设计语言的发展
- 程序设计的方法
- 程序开发的基本概念

什么是程序设计

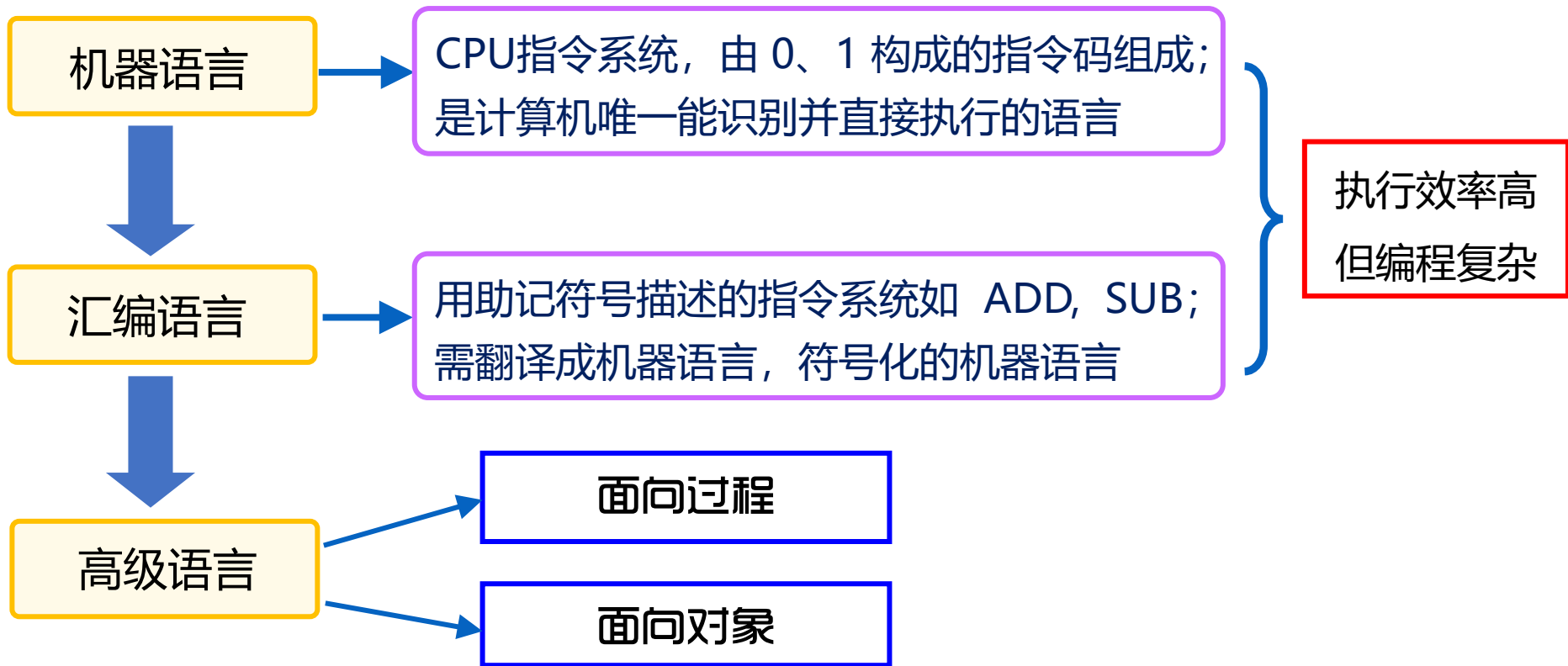
From Wikipedia, 2020

Computer programming is the process of **designing and building an executable computer program** to accomplish a specific computing result or to perform a specific task. Programming involves tasks such as: analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language.

The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem. Proficient programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.

❑ 程序设计：理解和分析问题，算法设计与分析，编程实现与调试

程序设计语言的发展



高级语言

高级语言

- 高级语言独立于机器，提高了语言的抽象层次
- 更接近人类自然语言，编程方便
- 不能直接被计算机识别，必须经过转换才能被执行

例： $x = x + y$; // 先计算 $x + y$ 的值，然后赋值给 x

两种转换方式

解释类语言

- ▶ 由解释器把源程序翻译成机器语言，每翻译一条执行一条，每执行一次就要翻译一次
- ▶ 优点：比较灵活，可以动态地调整、修改应用程序
- ▶ 典型代表：MATLAB、Python

编译类语言

- ▶ 由编译器将源程序编译成目标程序，然后生成可执行程序
- ▶ 可执行程序可以脱离语言环境独立执行，可重复运行，使用方便，一般执行效率高
- ▶ 典型代表：FORTRAN、C、C++

高级语言典型代表

❑ FORTRAN: Formula Translation

1956 年，由 IBM 的 J.W. Backus（哥伦比亚大学数学学士、硕士，图灵奖获得者）带领开发，高级语言诞生的标志，科学计算主流语言

❑ C

1972 年，由贝尔实验室的 D.M. Ritchie（哈佛大学数学博士，图灵奖获得者，UNIX 之父）开发，是一种通用的过程式编程语言，高效、灵活、功能丰富，主流的软件开发和科学计算语言

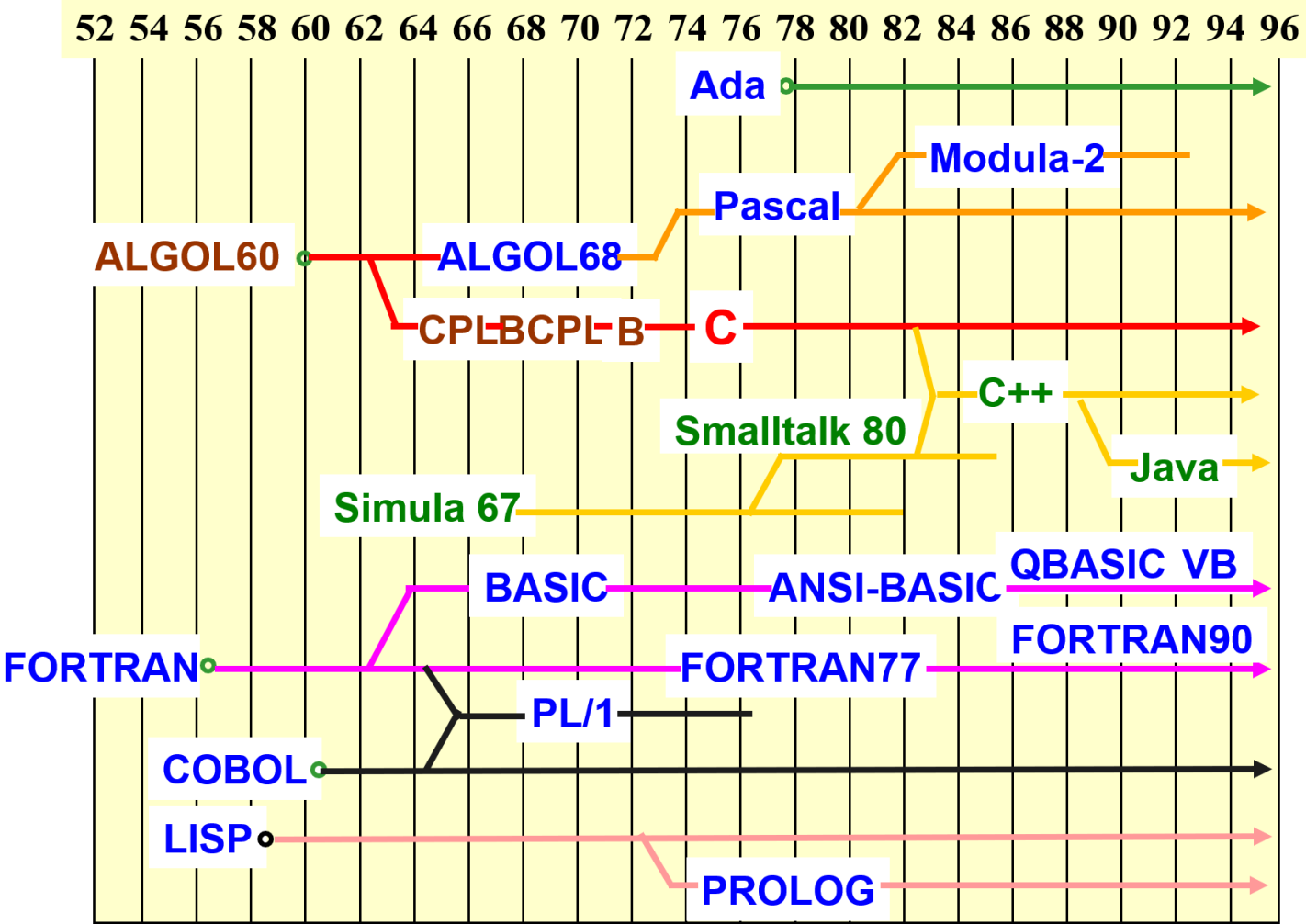
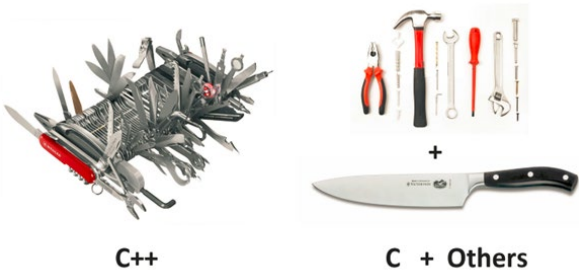
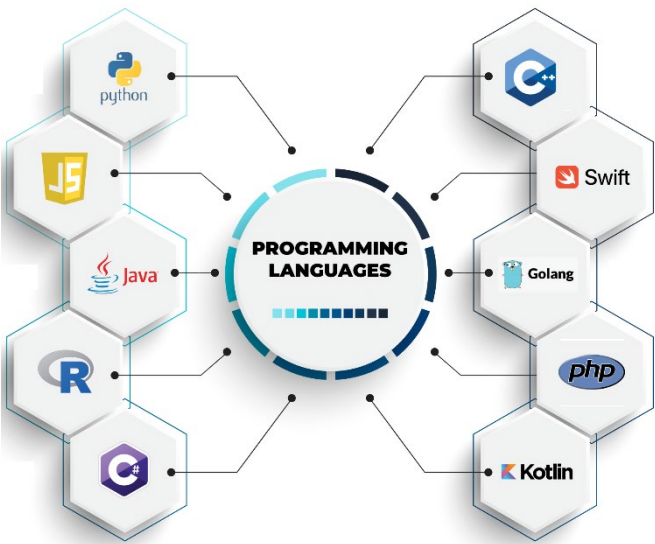
❑ C++

1983 年，由贝尔实验室的 B. Stroustrup 在 C 语言的基础上开发，引入并扩充了面向对象的概念功能

数学，特别是数学思维是计算机科学的一个支柱。—— B. Stroustrup

高级语言发展 (部分)

<https://www.levenez.com/lang/>



Sep 2026	Sep 2025	Programming Language
1	1	Python
2	3	C
3	2	C++
4	4	Java
5	5	C#
6	6	JavaScript
7	8	Visual Basic
8	11	SQL
9	13	R
10	18	Rust
11	12	Fortran
27		MATLAB

对比维度	 Python	 Java	 C++
 核心优点	✓ 语法极简、零基础快速上手，代码通俗易懂 ✓ 第三方库生态极强，AI 配套工具齐全 ✓ 开发效率高，不用纠结复杂底层语法，快速落地项目 ✓ 学习门槛低，自学无压力	✓ 稳定性强 ✓ 企业认可度高 ✓ 就业岗位多 ✓ 工程化体系成熟	✓ 运行性能极强 ✓ 底层控制力足 ✓ 硬件适配性好
 核心缺点	✗ 运行速度一般 ✗ 高性能大型项目适配较弱	✗ 语法繁琐冗余 ✗ 入门慢 ✗ 代码量大	✗ 学习难度高 ✗ 语法复杂 ✗ 上手极慢 ✗ 开发效率低
 适合学习人群	- 零基础小白 - 转行入门 - 学生党 - 想深耕人工智能 / 大模型的人群 - 零基础快速学技术人群	- 长期走职场后端开发 - 想进传统企业 - 计算机专业在校生	- 硬核技术深耕 - 嵌入式 - 游戏开发 - 竞赛算法爱好者
 应用场景	- 人工智能：深度学习 - CV 目标检测 - NLP 自然语言处理 - 大模型微调 - RAG 企业级落地 - 数据分析 - 自动化脚本 - 爬虫 - 简易后端	- 企业级后台系统 - 电商平台 - 政务金融大型项目 - 安卓开发	- 游戏开发 - 嵌入式硬件 - 自动驾驶底层 - 高性能精密开发

编程语言排行

Very Long Term History

<https://www.tiobe.com/tiobe-index/>

Programming Language	2026	2021	2016	2011	2006	2001	1996	1991	1986
Python	1	3	5	6	8	25	24	-	-
C	2	1	2	2	2	1	1	1	1
C++	3	4	3	3	3	2	2	2	6
Java	4	2	1	1	1	3	15	-	-
C#	5	5	4	5	7	13	-	-	-
JavaScript	6	7	7	10	9	9	20	-	-
Visual Basic	7	6	13	-	-	-	-	-	-

程序设计方法

- 面向 **过程** 的程序设计方法 (**过程式**、**结构化**、**模块化**)
- 面向 **对象** 的程序设计方法
- 基于 **模板** 的泛型编程思想

从面向过程入手，学习和理解面向对象和泛型程序设计

理解算法 → 实现算法 → 优化算法 → 设计算法

Procedure oriented programming, Functional Programming
Object-Oriented Programming, Generic Programming, Template Metaprogramming

面向过程程序设计

设计思路

- ❑ 自顶向下、分而治之，采用**模块分解**与功能抽象

程序结构

- ❑ 按功能划分为若干个基本模块
- ❑ 各模块间的关系尽可能简单，功能上相对独立
- ❑ 其模块化实现的具体方法是使用函数/子程序

优点

- ❑ 能有效将复杂任务分解成多个易于控制和处理的子任务，便于系统开发和维护

面向过程程序设计

不足之处

- ❑ 数据和处理数据的过程相互独立：当数据结构改变时，所有相关的处理过程都要进行修改
- ❑ 随着程序规模不断扩大，模块数呈指数级递增，模块间的数据传递五花八门，同一程序中模块之间的**关系错综复杂**，结构化程序设计方法对程序的可维护性和重用性越来越力不从心

软件危机

落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

面向对象程序设计

- ❑ 将数据及对数据的操作方法封装在一起，作为一个整体（ **对象** ）
- ❑ 对同类型对象抽象出其共性，形成 **类 (class)**
- ❑ 类通过一个简单的外部接口，与外界发生关系
- ❑ 对象与对象之间通过消息进行通信

面向对象程序设计的特点

- ❑ 程序模块间的关系更为简单，程序模块的独立性、数据的安全性就有了良好的保障
- ❑ 通过继承与多态，可以大大提高程序的可重用性，使得软件的开发和维护都更为方便

程序开发基本概念

- ❑ 源程序/源代码、目标程序/目标代码、可执行程序
- ❑ 翻译程序/编译器：汇编、解释、编译

C++ 程序开发过程

- ❑ 编写源程序：C++ 源程序的后缀名为 `.cpp`
- ❑ 编译：生成目标程序
- ❑ 连接：将目标程序和库文件连接生成一个可执行文件，比如 `.exe`
- ❑ 运行、调试



- 计算机基础：信息表示与存储，算法
- C++ 面向过程的程序设计
- C++ 面向对象的程序设计
- 常见算法设计（排序、数值算法、...）

面向过程 的程序设计

- C++基本元素：字符集，词汇，保留字
- 基本数据类型，变量，常量
- 表达式与语句
- 算法与三种基本结构
- 数组
- 引用，指针，字符串
- 函数，递归
- 简单的文件输入输出
- 预处理与多文件工程

面向对象 的程序设计

- 类与对象
- 结构体与联合体
- 数据的共享与保护
- 对象的生存期，静态成员
- 友函数与友元
- 继承与派生、访问控制
- 多态性、运算符重载
- 文件操作
- 标准模板库和泛型编程



学习建议

- ❑ 养成良好的编程习惯和编程风格（强制 → 自然）
书写格式，命名规则，充分的注释，...
- ❑ 注重提升编程质量和编程效率（模仿 → 创造）
结构设计，功能分解，条理清晰，...
- ❑ 提升：数据结构与算法，操作系统，计算机网络，... ..

重基础，多练习，勤思考

“Programming is learned by writing programs”

—— Brain Kernighan



谢谢



Programming