

CS100 Introduction

从写代码到解决问题

Speaker Chaofan Li (lichf2025@shanghaitech.edu.cn)

ShanghaiTech University



上海科技大学

ShanghaiTech University

目录

1 吃水不忘挖井人

2 认识你的 TA

3 我们为什么学习编程?

吃水不忘挖井人

革命尚未成功

同志仍須努力

孫文



GKxx：现代 CS100 的重要建设者

上海科技大学 2020 级计算机科学与技术专业学生，曾多次担任 CS100 的 TA。

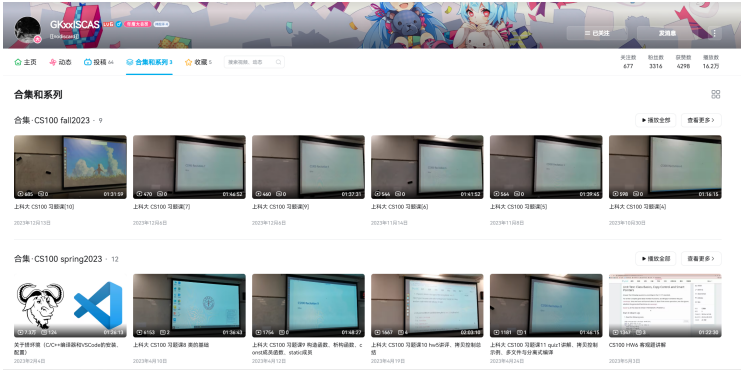
改良课件与教学方式，改进课程考核，为今天的 CS100 打下重要基础。

如今已经“飞升”国科大／中国科学院软件研究所。

致敬，也接力。把课讲清楚，把工具做好，把这份认真传下去。

Bilibili：**GKxxISCAS**　　GitHub：**GKxxUCAS**

想成为“语言律师”？去看 GKxx 的课



- 想深入钻研 C/C++ 的语言细节，可以去 Bilibili 看**GKxxISCAS**的 CS100 讲课视频。
- **环境配置请看我们的新版课件**：习题课零（Environment Setup），不要照搬旧视频中的配置步骤。

认识你的 TA

HeadTA：李超凡



邮箱：lichf2025@shanghaitech.edu.cn

2021 届本科生／ 2025 届博士研究生
GKxx**关门大弟子**　CET6 备战中
娄鑫课题组招新大使

助教经历

- 2022 Fall 至今：信导助教
- 2024–2026 Fall：CS100 HeadTA
- 2026 Spring：CS110 优秀 TA

竞赛与带队

- 2023 届 iGEM 全球前十、金奖
- 2023 年电赛二等奖
- 2025 年金牌电赛教练

TA：李宇轩



GitHub：**liyuxuan3003**

邮箱：liyx22025@shanghaitech.edu.cn

2021 届上海大学微电子专业优秀毕业生

2025 届娄组工程博士　3003 年出生

助教经历

- 2025 Fall、2026 Spring：信导助教
- 2026 Spring：CS110 计算机体系结构助教

竞赛经历

- 2023 全国大学生集成电路创新创业大赛：**全国二等奖**
- 2023 全国大学生 FPGA 创新设计竞赛：**全国一等奖**

Windows 与 Verilog 的**掘墓人**

LaTeX 古法 TikZ 的**非遗传承者**

Spinal、Linux、LaTeX、Vim 宣传大使

李宇轩的开源笔记带货

GitHub：**github.com/liyuxuan3003**

计算机体系结构	**ComputerArchitecture**
数字电路	**DigitalCircuit**
模拟电路	**AnalogCircuit**
电路基础	**ElectricCircuits**
电磁场与电磁波	**ElectromagneticField**
固体物理	**SolidPhysics**
半导体物理	**SemiconductorPhysics**
大学物理	**CollegePhysicsNew**
微积分	**Calculus**
数学物理方法	**MathematicalPhysics**

这些开源笔记，没有一份是 AI 写的。

欢迎**Star**笔记仓库、**Follow**作者，支持非遗传统手艺人！

立志成才 报国裕民

TA：蒋烨



邮箱：jiangye2026@shanghaitech.edu.cn

2022 级优秀本科生

娄组 2026 级工学博士　研究方向：**3DGS**

二次元浓度**很高**

最好的御主、教主、司辰、指挥官、sensei、博士、管理员







TA：王诗涵



邮箱：wangshh2023@shanghaitech.edu.cn

娄组预备工程博士

2025 年全国大学生电子设计大赛**省一等奖**

23 级四驱兄弟中**唯一留得住的区**

乐邦詹士（**一人一组版**）

下早攻，只把天赋带给**娄组**

自动挡科目三**挂了三次**

进娄组都会失去一点什么的

我失去了驾照，超凡失去了六级。

TA：刘文杰



邮箱：liuwj2026@shanghaitech.edu.cn

张芯韵组

2026 级电子科学与技术学术研究生

TA：孟祥琨



邮箱：mengxk2026@shanghaitech.edu.cn

本科：同济大学

张芯韵组

2026 级电子科学与技术学术研究生

怎样让 TA 更快帮到你？

“我的代码不行”很难定位；一个**可以复现的问题**更容易得到有效帮助。

1. 我想做什么？题目要求、输入与预期输出是什么？
2. 实际发生了什么？提供最小代码、实际输出和完整报错。
3. 在什么环境发生？系统、编译器版本、编译与运行命令。
4. 我已经尝试过什么？哪些修改改变了现象？

优先提供可复制的代码与文字，让别人能在自己的环境中重现问题。

我们为什么学习编程？

编程要解决什么问题？

编程是把一个需求，变成计算机可以**重复、准确地执行的过程**。

- 自动化：把一千份文件按规则整理，而不是手动点一千次。
- 数据处理：从一批观测中提取信息，而不是只盯着原始数字。
- 建模与计算：把现实问题抽象成数据、约束与算法。
- 构建系统：让多个功能组合起来，持续、可靠地工作。

语言是表达方案的工具。真正需要你决定的是：**做什么、怎样做、怎样证明做对了**。

从一句需求，到一个明确的问题

需求：“帮我算一下全班的平均成绩。”

1. **明确输入**：成绩从哪里来？一共有多少人？是否合法？
2. **约定规则**：缺考如何处理？空名单怎么办？保留几位小数？
3. **设计步骤**：读取数据，检查约束，累加求和，计算并输出。
4. **验证结果**：用能手算的小例子和边界情况检查程序。

计算机不会替你补全含糊的需求，AI 生成代码也不能替你决定这些规则。

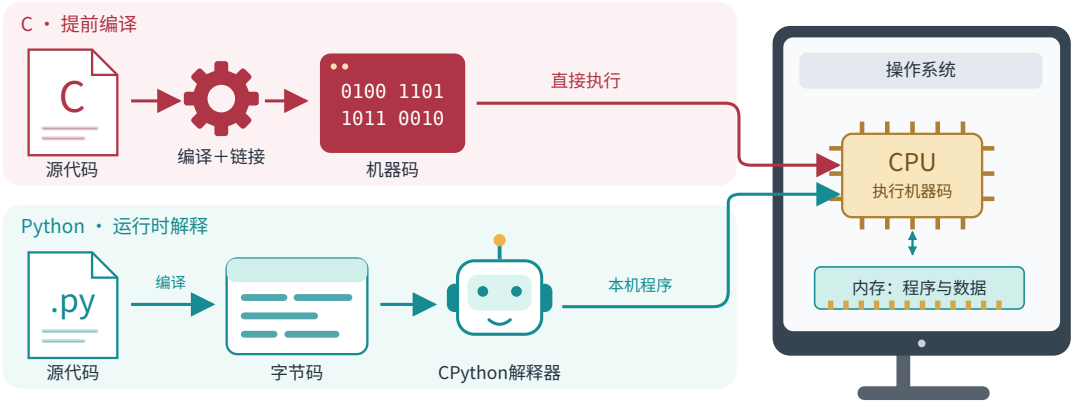
为什么还要学 C 语言？

C 让许多被高级抽象隐藏的细节，成为你必须理解的对象。

- **数据表示**：类型、取值范围与精度，都会影响计算结果。
- **内存模型**：数组、地址、指针、对象的生存期。
- **程序执行**：源文件怎样经过编译与链接，变成可执行程序。
- **资源与代价**：一次复制、一次分配、一个循环意味着什么。

学 C 帮助我们建立程序如何运行的基本模型，为后续系统课程打基础。

C 和 Python：程序怎样跑起来？



字节码交给解释器，机器码交给CPU · 以CPython为例

一个比喻：全能翻译官与译好的书

Python：带着一位全能翻译官

- 这位“翻译官”就是**解释器**，在程序运行时帮助执行代码。
- 去中国讲中文，去法国讲法语：各个平台有适配自己的解释器。
- 同一份代码通常可以跨平台运行，前提是解释器和依赖兼容。

C：一本已经译成中文的书

- **编译器**提前完成翻译，得到目标平台的可执行程序。
- 中国读者直接读，不必现场翻译，通常执行效率较高。
- 拿给不懂中文的法国读者就看不懂：换平台往往需要重新编译。

这里的“中文书”指**编译后的程序**。C 源代码仍可移植；能否直接运行，取决于操作系统、处理器架构等条件。

立志成才 报国裕民

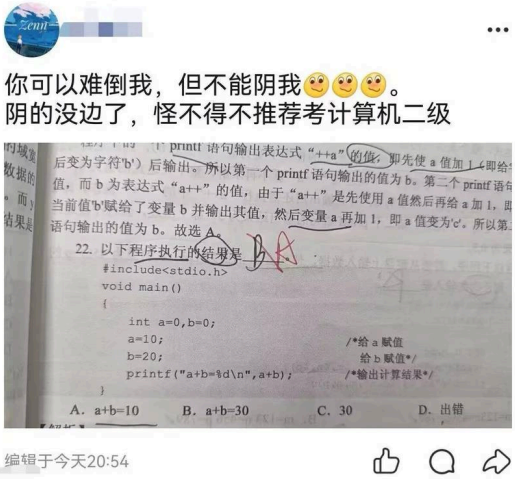
我们不做“语言律师”

不以背诵冷僻规则、猜刁钻表达式为目标。

把正确性相关的规则弄清楚：

- 类型转换与整数除法
- 数组越界与变量初始化
- 对象生命周期与未定义行为

写清晰、可解释的代码；
不确定时，**查文档、做验证**。



AI 时代，哪些能力更重要？

AI 可以帮你写代码，你仍然需要：

1. **定义问题**：说清输入、输出与约束。
2. **审查方案**：判断算法与复杂度是否合适。
3. **验证行为**：用测试检查结果。
4. **承担责任**：能解释、修改和维护代码。

一个简单的自检

明天代码出错，你能从哪里开始查？



把 AI 放进有反馈的学习过程

1. 先自己写出问题描述、思路和几个预期结果。
2. 请求提示、解释或方案比较，并指出自己卡在哪里。
3. 逐段读懂代码，在统一环境中编译并检查诊断信息。
4. 用正常输入、边界输入与反例验证，发现问题后再修改。

编译成功不等于逻辑正确，几个样例通过也不等于程序正确。

作业与考试能否使用 AI、如何说明使用情况，以当次课程要求为准。

为什么 CS100 要规范开发方式？

“在我电脑上能跑”，还不足以让别人复现你的结果。

- **统一编译环境**：减少系统、语言标准和编译选项造成的差异。
- **统一代码习惯**：让同学、TA 和未来的自己都能读懂代码。
- **使用版本管理**：知道改了什么，保留可运行的版本。
- **保留验证过程**：提交的结果可以被重新构建和检查。

这些习惯从第一个小程序就开始建立，不必等到项目变大。

编译环境：让构建过程可复现

本课程统一使用**C17 和 C++17**；习题课沿用 r0 的 Ubuntu ／ WSL 与 GCC 路线。

```
gcc --version
gcc -std=c17 -Wall -Wextra -Wpedantic main.c -o main
g++ -std=c++17 -Wall -Wextra -Wpedantic main.cpp -o main
./main
```

- 显式指定语言标准，避免依赖编译器的默认设置。
- 开启常用警告，逐条理解并修复，不把警告当作背景噪声。
- 记录系统、编译器版本和命令；编辑器可以不同，构建过程要清楚。

C 源文件使用 gcc，C++ 源文件使用 g++。编译器具体版本及其他评测参数，以课程公告为准。

代码习惯：写给人读，也交给机器执行

- **命名表达含义**：用 `studentCount` 表达人数，避免一串无意义缩写。
- **格式保持一致**：统一缩进、括号和空格，让结构一眼可见。
- **拆分职责**：一个函数完成一件清楚的事，减少重复与隐含依赖。
- **注释解释原因**：说明约束、假设和设计选择，不逐句翻译代码。
- **尊重题目接口**：严格遵守输入输出格式，不夹带调试信息。

代码越容易读懂，就越容易发现错误、验证行为和继续修改。

命名有约定：小驼峰、大驼峰与成员变量

- **小驼峰**：变量、参数与函数。
  `studentCount`、`studentId`、`getId`
- **大驼峰**：class 类名。
  `Student`
- **成员变量**：加 `m_` 前缀。
  `m_id`、`m_studentCount`

```cpp
class Student {
public:
    explicit Student(int studentId)
        : m_id(studentId) {}

    int getId() const { return m_id; }

private:
    int m_id;
};
```

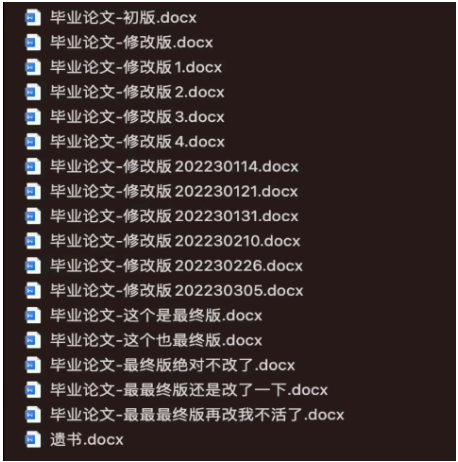
右侧为 C++17 示例。命名是**团队约定**，遵循课程或项目规范，并保持一致。

版本管理：不要再用“最终版 2”

Git 保存每次有意义的修改，方便回看。

```
git init
git status
git diff
git add main.c
git diff --staged
git commit -m "Handle empty input"
git log --oneline
```

- 一次提交一件事，写清修改含义。
- 提交源码与配置，用 `.gitignore` 排除生成文件。
- 可完全本地使用；远端备份须遵守作业公开范围要求。



一次小修改，也值得走完这个循环

明确问题 → 编写代码 → 编译检查

测试验证 → 检查差异 → 提交版本

- **正常情况**：输入几条普通成绩，与手算结果对照。
- **边界情况**：只有一条记录、零分、满分，是否符合约定？
- **特殊情况**：空输入与不合法数据，按题目约定处理。
- **回归检查**：修好一个问题后，原来正确的情况是否仍然正确？

每次只改清楚的一件事，你会更容易知道：这次为什么变好了，哪里又出了问题。

我们希望你带走什么？

1. 能把模糊需求转化为明确的问题与可执行的步骤。
2. 能解释程序的行为，理解数据、内存与计算的基本模型。
3. 能使用工具定位问题，并用测试验证自己的判断。
4. 能留下清楚的代码、可复现的环境与可追溯的修改记录。

从 CS100 开始

让代码值得信任，让自己有能力判断它为什么值得信任。

立志成才 报国裕民

Thank You