

Creative Development

AP Computer Science Principles ▪ Unit 1

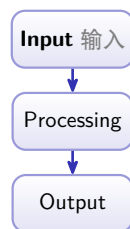
AP Computer Science Principles



Creative Development

What we will cover

- 1 Collaboration
- 2 Program function & purpose
- 3 Design & development
- 4 Finding & fixing errors



1 Collaboration

Software is a team sport

A large **program** 程序 is designed, written, tested & fixed by a team. **Collaboration** 协作 means working together toward a shared goal.

Driver 驾驶员
types the code

Naviga
watches,
catche



*Software is designed, written, and tested
by people*

Pair programming 结对编程: two people, one computer, swapping roles. Teams merge work with **version control** 版本控制 and explain it with **documentation** 文档.

2 Function & purpose

Purpose, function & the IPO model

Purpose 目的 = **why** a program exists; **function** 功能 = **what** it does.

Every program fits **input** → **processing** → **output** (the IPO model).

Many are **event-driven** 事件驱动: they wait and react to **events** 事件 (a click) via an **event handler** 事件处理程序.



3 Design & development

Creative Development
└ Design & development

The iterative development cycle

The process is **iterative** 迭代的, **not** a straight line: investigate, design, implement, test – **then repeat**.

Development is **incremental**: build and test a **small piece**, then add the next.

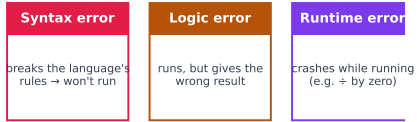


Development is iterative and collaborative

Break a large problem into pieces (**decomposition** 分解); plan with a **program requirement** 需求 and **pseudocode** 伪代码; keep it readable with a **comment** 注释.

4 Errors

Four kinds of error



Syntax error 语法错误 – breaks the language rules; won't run

Logic error 逻辑错误 – runs, but wrong answer

Runtime error 运行时错误 – crashes mid-run (divide by zero)

Overflow error 溢出错误 – value too large for its storage

Finding & fixing bugs

A **bug** 缺陷 is a mistake; **debugging** 调试 is finding & fixing it.

test many inputs,
incl. **edge cases** 边界情况

compare with
the **expected output** 预期输出

divide & check; re-test after each fix

A logic error runs with no crash – "it ran" is not "it's correct". Only comparing output to what you expected catches it.

Hand-tracing with a trace table

A **trace table** 追踪表 records **each variable's value** as the program runs – line by line, by hand.

It is how you find a **logic error** that no crash will ever reveal.

count	total	output
1	5	
2	12	
3	20	20

each variable, at each step

Worked example – a logic error hiding in plain sight

A program should print the average of two numbers, but runs $\text{avg} = a + b / 2$

Order of operations: $/$ runs **before** $+$, so it computes $a + \frac{b}{2}$ – not the average. Test with $a = 4$, $b = 6$: the bug gives $4 + 3 = 7$; the fix $\text{avg} = (a + b) / 2$ gives $\frac{10}{2} = 5$.

Testing with **known inputs** is exactly how you find **and confirm** a logic error – the program never crashed.

Key points

Collaboration + version control build better software

Every program is **input**
→ **processing** → **output**

Development is **iterative**:
plan, prototype, test, refine

Syntax, logic, run-time & overflow **errors**

Check yourself

- 1 $\text{avg} = a + b / 2$ runs and prints a number. Which error is it?
- 2 Which error stops the program from running at all?
- 3 Name the three parts of the IPO model.
- 4 What does the navigator do in pair programming?

Answers 1. a logic error 2. a syntax error 3. input, processing, output 4. watches, thinks ahead, catches mistakes