

Algorithm Design

AS Computer Science ■ Topic 9

A-Level Computer Science



Algorithm Design

What we will cover

- 1 Computational thinking
- 2 Algorithms
- 3 Pseudocode constructs
- 4 Notations & refinement
- 5 Logic statements
- 6 Recap



1 Computational thinking

Algorithm Design

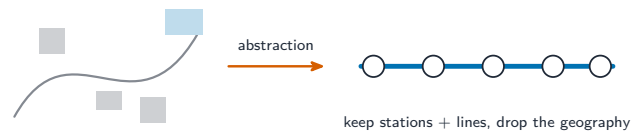
└ Computational thinking

Abstraction

Abstraction 抽象 keeps the **essential** features and drops the irrelevant detail. A metro map keeps the stations and lines, not the geography.

real geography

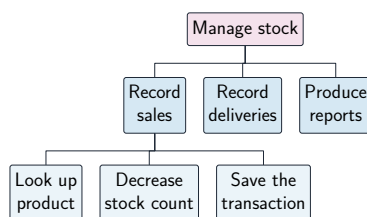
metro map



Algorithm Design

└ Computational thinking

Decomposition



Decomposition 分解 breaks a big problem into smaller sub-problems, each easy to solve. Each module becomes a **procedure** 过程 or function.

Algorithm Design

└ Computational thinking

Algorithms and the identifier table

An **algorithm** 算法 is a sequence of steps that is **unambiguous**, **deterministic**, **finite** and **effective**. It says *what* to do – independent of the language.

Name	Type	Description
ItemCost	REAL	cost
InStock	BOOLEAN	in stock?
SaleDate	DATE	sold on

An **identifier table** 标识符表 names every variable first.

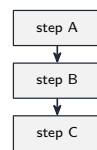
2 Pseudocode

Algorithm Design
└ Pseudocode

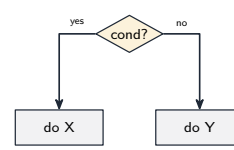
The three basic constructs

- **sequence** 顺序 – one step after another
- **selection** 选择 – IF/ELSE, CASE
- **iteration** 迭代 – FOR, WHILE, REPEAT

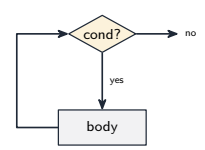
sequence



selection



iteration



Algorithm Design
└ Pseudocode

Pseudocode operations

Selection

```

IF Age >= 18 THEN
  OUTPUT "Adult"
ELSE
  OUTPUT "Minor"
ENDIF
  
```

- assignment: $x \leftarrow 5$
- DIV (integer $\div$), MOD (remainder)
- WHILE tests before; REPEAT tests after
- & joins strings (concatenation 拼接)

Algorithm Design
└ Pseudocode

Worked example: trace the algorithm

```

x ← 0
FOR i ← 1 TO 5
  IF i MOD 2 = 0 THEN
    x ← x + i
  ENDIF
NEXT i
OUTPUT x
  
```

i	i MOD 2	x
1	1	0
2	0	2
3	1	2
4	0	6
5	1	6

What is the output?

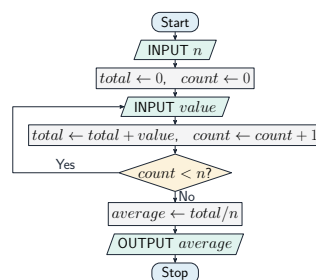
output: 6

Fill the **trace table** 跟踪表 row by row – never run the whole loop in your head. Only even i (2 and 4) change x.

3 Notations and refinement

Algorithm Design
└ Notations and refinement

Three notations



One algorithm, three ways:

- **structured English** 结构化英语 – high-level
- **flowchart** 流程图 – standard shapes
- **pseudocode** 伪代码 – closest to code

Each IF = a diamond; each loop = a back-arrow.

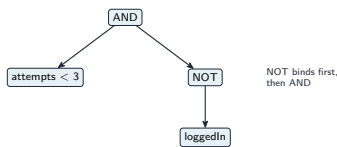
Stepwise refinement



Stepwise refinement 逐步求精 starts with a high-level outline and **expands each step** until it can be coded. "Compute the average" → a loop that sums, then divides by n .

4 Logic

Logic statements



A **Boolean** 布尔 condition controls branching. **Precedence** 优先级: NOT > AND > OR.
■ write $a = 1$ OR $a = 2$ (not $a = 1$ OR 2)
■ **De Morgan** 德摩根定律: NOT (A AND B) = (NOT A) OR (NOT B)

5 Recap

Topic 9 – key takeaways

1 Thinking

abstraction keeps essentials; decomposition splits

2 Constructs

sequence, selection, iteration

3 Notations

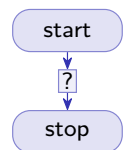
structured English, flowchart, pseudocode

4 Logic

NOT > AND > OR; De Morgan

Check yourself

- 1 Which technique drops irrelevant detail?
- 2 Which loop always runs at least once?
- 3 Flowchart shape for a decision?
- 4 Simplify NOT (A AND B).



Answers 1. abstraction 2. REPEAT...UNTIL 3. a diamond 4. (NOT A) OR (NOT B)