HOLIDAY HOMEWORK 假期作业

AS-Level Computer Science

Exercise sheets and past-paper practice, topic by topic.

每个单元：练习卷 + 历年真题。

For class or self-study • no answer keys included.

Contents 目录

1 Information representation – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
bit	位			
byte	字节			
binary	二进制			
denary	十进制			
number system	数制			
place value	位值			
hexadecimal	十六进制			
nibble	半字节			
remainders	余数			
two's complement	补码			
sign bit	符号位			
overflow	溢出			
BCD	二进码十进数			
character set	字符集			

Recall

You already started this at IGCSE. Bring it with you:

- A **bit** 位 is a single 0 or 1. Eight bits make one **byte** 字节.
- You used **binary** 二进制 (base 2) and **denary** 十进制 (base 10, the everyday numbers), and converted between them.
- At AS we add three new ideas: **hexadecimal**, **negative** numbers in binary, and **BCD**.

Nothing here is harder than IGCSE — it builds straight on the binary you already know.

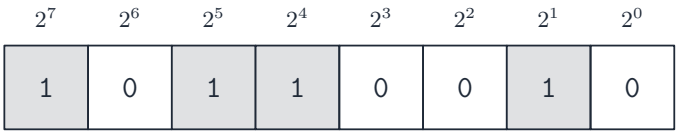
New ideas

Read these first. Each idea has a worked example. Then do the Practice below.

■ 1 Number bases and the general method

A **number system** 数制 uses a fixed set of digits, and a digit's value depends on its **place value** 位值 (which column it is in). Binary is base 2, denary base 10, and **hexadecimal** 十六进制 base 16. Four bits make a **nibble** 半字节. The two methods below work for **any** base — learn these, not a binary-only trick.

Any base → denary. Multiply each digit by the base raised to its position (count positions from 0 at the right), then add.

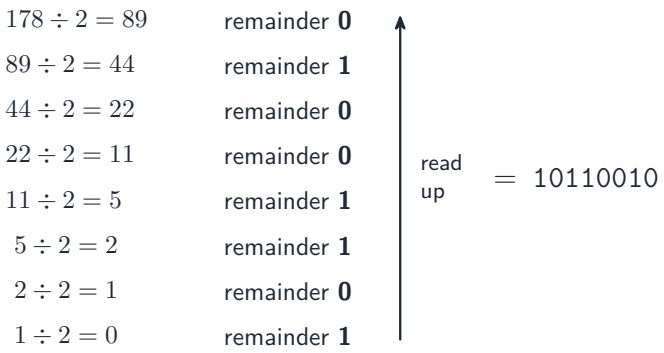


$$(1\times2^7)+(1\times2^5)+(1\times2^4)+(1\times2^1)$$
$$= 128+32+16+2 = \mathbf{178}$$

$$10110010_2 = (1\times2^7)+(1\times2^5)+(1\times2^4)+(1\times2^1) = 128+32+16+2 = 178.$$

For hexadecimal you use powers of 16 instead — exactly the same method.

Denary → any base. Divide by the base again and again; the **remainders** 余数, read from bottom to top, are the digits.



To get hexadecimal instead, divide by 16 the same way.

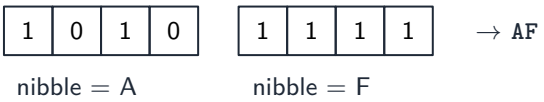
■ 2 Working with hexadecimal

Hexadecimal digits are 0–9, then A, B, C, D, E, F for 10 to 15. It is a short way to write binary, so programmers use it for memory addresses and colour codes.

Hex → denary is the same powers method, with base 16:

$$0\text{xB2} = (11\times16^1) + (2\times16^0) = 176 + 2 = 178.$$

Shortcut (binary ↔ hex): group the bits into nibbles of four from the right, and convert each nibble on its own.




```
1011 0010   ->   B   2      so 10110010 = 0xB2
```

■ 3 Negative numbers: two's complement

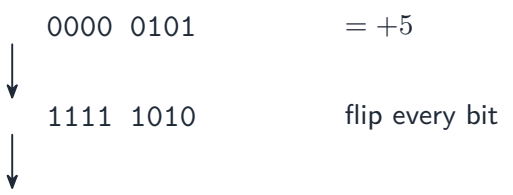
To store a negative whole number we use **two's complement** 补码. The left-most bit becomes the **sign bit** 符号位: 0 means positive, 1 means negative.

To write a negative number (8 bits): write the positive value, flip every bit, then add 1.

Worked example — store −5:

step	bits
+5	0000 0101
flip every bit	1111 1010
add 1	1111 1011

So −5 is 11111011. If a sum needs more bits than the register has, you get **overflow** 溢出 and the answer is wrong.



Watch out: flip the bits *then add 1* — just once. Flipping a second time only takes you back to where you started.

4 Binary Coded Decimal (BCD)

In **BCD** 二进码十进数 each denary digit is written as its own 4-bit nibble. It keeps the digits separate (useful for clocks and calculators).

Worked example: 59 in BCD is 0101 1001 — that is 5 then 9, **not** the pure binary 00111011.

5 Text is data too

Letters are stored with a **character set** 字符集 — a table giving each symbol a number (for example, ASCII stores `A` as 65). The computer still stores only bits; the character set says what they mean.

Practice

Try these in order. The methods are all above. Show your working.

2.1 Convert the denary number 45 to an 8-bit binary number. [2]

2.2 Convert the binary number 00110100 to denary. [1]

2.3 Convert the binary number 11101010 to hexadecimal. [2]

2.4 Convert the hexadecimal number 2F to denary. [1]

2.5 Using 8-bit two's complement, write the number −12. Show each step. [3]

2.6 Write the denary number 72 in BCD. [2]

2.7 Give one reason programmers use hexadecimal instead of long binary numbers. [1]

2.8 In 8-bit two's complement, explain *why* the left-most bit alone tells you whether the number is negative. [2]

7 Complete the binary addition. Show your working.

$$\begin{array}{r} 1\,0\,0\,1\,1\,1\,1\,0 \\ 0\,1\,1\,0\,0\,0\,0\,1 \\ +\,0\,0\,0\,1\,1\,0\,0\,1 \\ \hline \end{array}$$

[3]

1 Information representation – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
bitmap	位图			
pixel	像素			
image resolution	图像分辨率			
colour depth	颜色深度			
vector graphic	矢量图形			
sampling	采样			
sampling rate	采样率			
Compression	压缩			
lossless	无损			
lossy	有损			
run-length encoding	行程编码			

Recall

In **Part 1** you represented numbers (binary, denary, hexadecimal, two's complement, BCD) and text (character sets). Part 2 looks at how **images and sound** are stored as numbers, and how files are made smaller.

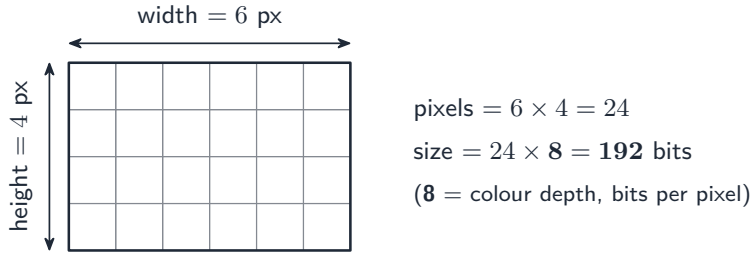
New ideas

■ 1 Bitmap images (and vectors)

A **bitmap** 位图 image stores the colour of every **pixel** 像素 in a grid. Two settings decide its quality and size:

- **image resolution** 图像分辨率 — the width × height in pixels;
- **colour depth** 颜色深度 — the number of bits per pixel (1 bit → black/white, 8 → 256 colours, 24 → "true colour").

$$\text{file size (bits)} = \text{width} \times \text{height} \times \text{colour depth}.$$

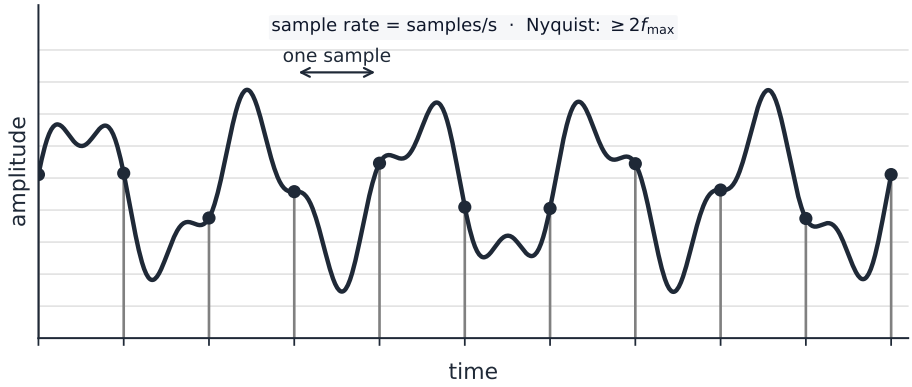


A **vector graphic** 矢量图形 instead stores *instructions* to draw shapes (with properties like colour and position). It scales to any size without blur — great for logos — but cannot store a photograph.

Watch out: doubling the resolution does not double the file size — the width *and* the height both double, so the size goes up about *four* times.

■ 2 Sound

Sound is **analogue** (a smooth wave). To store it the computer takes **sampling** 采样 — reading the wave's height at regular time steps. The **sampling rate** 采样率 is the number of samples per second (CD quality is 44.1 kHz).



A higher sampling rate (or more bits per sample) gives better quality but a bigger file.

3 Compression

Compression 压缩 makes a file smaller, saving storage and transmission time. Two kinds:

- **lossless** 无损 — the original is recovered **exactly** (text, programs, PNG, ZIP);
- **lossy** 有损 — some detail is dropped for a **much smaller** file (JPEG, MP3, streamed video).

One lossless method is **run-length encoding** 行程编码 (RLE): store "n copies of x" instead of repeating x — good for big flat areas.

lossless — recovered exactly (RLE, PNG, ZIP)
lossy — smaller, some detail lost (JPEG, MP3)

Practice

1.1 A bitmap image is 100 pixels wide, 100 high, with a colour depth of 24 bits. Find its file size in bits, then in bytes. [3]

1.2 State whether a bitmap or a vector graphic is better for a company logo, and give one reason. [2]

1.3 State the effect on (a) the quality and (b) the file size of using a higher sampling rate. [2]

1.4 State whether lossless or lossy compression should be used for (a) a program file, (b)

streaming a video. [2]

1.5 Use run-length encoding to compress the pixel row `W W W W B B W W` (write your code as count–value pairs). [2]

1.6 Explain *why* a photograph cannot be stored well as a vector graphic. [2]

7 A student takes a photograph of a science experiment.

(a) The photograph is saved as a bitmapped image.

(i) Define the following bitmap terms.

Colour depth .....

.....

File header .....

.....

[2]

(ii) Explain why changing the image resolution will affect the image quality and file size.

Image quality .....

.....

.....

File size .....

.....

.....

[2]

(iii) Identify **one** lossless method of compressing an image.

..... [1]

(b) The student draws a picture on paper that is scanned into the computer and saved as a vector graphic.

Define the vector graphic terms property and drawing list.

Property .....

.....

Drawing list .....

.....

[2]

6 A computer system stores text, images and sound.

(a) A character set is used to represent characters in a computer.

Identify **and** describe **one** character set.

Character set .....

Description .....

.....

[2]

(b) The colour of each pixel in a bitmapped image is represented by 8 bits.

(i) State the largest number of different colours that can be represented by 8 bits.

..... [1]

(ii) State **one** drawback of increasing the number of bits that represents each pixel in the bitmap image.

.....

..... [1]

(iii) A bitmap image can be compressed using lossy compression.

Explain the reasons why lossy compression is often suitable for a bitmap image.

.....

.....

.....

..... [2]

(c) (i) Explain how an analogue sound wave is converted into digital data.

.....

.....

.....

..... [2]

(ii) Describe **one** method of compressing a sound file using lossy compression.

.....

.....

.....

..... [2]

2 Communication – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
network	网络			
local area network	局域网			
wide area network	广域网			
bandwidth	带宽			
server	服务器			
client	客户端			
peer-to-peer	对等网络			
topology	拓扑			
packet	数据包			
protocol	协议			

Recall

At IGCSE you used networks. Bring this with you:

- A **network** 网络 connects computers so they can share data and resources.
- You met the internet and wireless connections.

At AS we look at how networks are arranged and the rules that let devices talk to each other.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

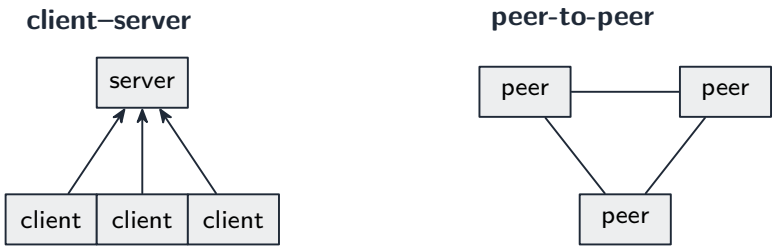
1 Types of network

- A **local area network** 局域网 (LAN) covers one site — a school or an office.
- A **wide area network** 广域网 (WAN) joins sites over long distances; the internet is the largest WAN.
- The **bandwidth** 带宽 of a link is how much data it can carry each second.

Worked example. The computers in one school form a LAN; when the school connects to a bank's computers across the country, that long-distance link is part of a WAN.

2 Client–server and peer-to-peer

- In a client–server network a powerful **server** 服务器 stores files and services, and many **client** 客户端 machines request them.
- In a **peer-to-peer** 对等网络 network the computers share directly, with no central server.

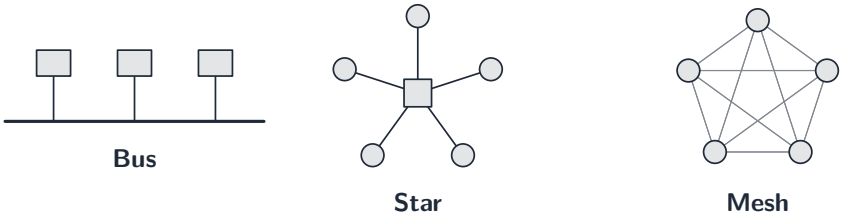


Worked example. A school's shared drive is client–server (one server holds the files); two phones sending a photo straight to each other is peer-to-peer (no server in between).

Watch out: the internet is a WAN, not a LAN — and "the web" (web pages) is just one *service* that runs on the internet, not the internet itself.

■ 3 Topologies and protocols

The **topology** 拓扑 is the shape of the connections — for example a *bus* (one shared line) or a *star* (every device joined to a central switch).



Data travels in small **packet** 数据包 units. A **protocol** 协议 is an agreed set of rules that lets different devices communicate — for example TCP/IP on the internet.

Worked example. On a star network each computer has its own cable to the switch, so if one cable breaks, only that one computer drops off and the rest keep working.

Practice

2.1 State what is meant by the bandwidth of a connection. [1]

2.2 State one difference between a LAN and a WAN. [2]

2.3 Give one difference between a client–server network and a peer-to-peer network. [2]

2.4 Explain what a protocol is, and give one example. [2]

2.5 Give one advantage of a star topology. [1]

2.6 Explain *why* a star topology keeps working for the other computers when one cable

fails, but a bus can fail for everyone. [2]

2.7 Challenge. A small office of five computers wants to share files but cannot afford a dedicated server. Suggest which network type suits them and give a reason. [2]

2 Communication – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
protocol	协议			
World Wide Web	万维网			
domain name	域名			
Domain Name System	域名系统			
Ethernet	以太网			
twisted-pair	双绞线			
fibre-optic	光纤			
network interface card	网络接口卡			
switch	交换机			
router	路由器			

Recall

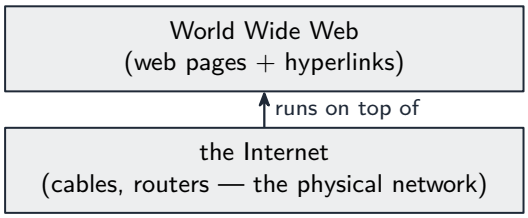
In **Part 1** you met LAN vs WAN, client–server vs peer-to-peer, network topologies, bandwidth, and protocols. Part 2 looks at the internet and the web, how a name becomes an address, wired vs wireless links, and the hardware that joins a LAN.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 The internet and the World Wide Web

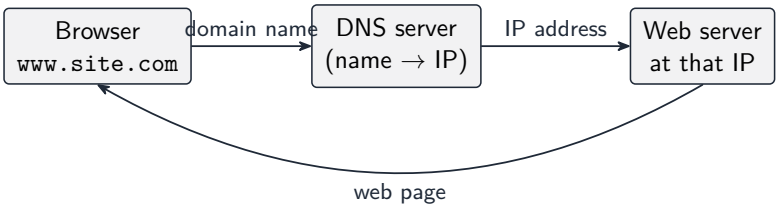
The **internet** is a global *network of networks* that all use the same **protocol** 协议 family (TCP/IP). The **World Wide Web** 万维网 is just one service running on it — linked pages, found by their address (URL) and viewed in a browser. Email and file transfer are other internet services, not part of the web. Every device has an IP address (IPv4 is four numbers 0–255, e.g. `192.168.1.10`).



Worked example. When you send an email and open a web page, both travel over the internet — but only the web page is part of the World Wide Web; email is a separate service.

■ 2 Names and addresses (DNS)

A web address contains a **domain name** 域名 (like `example.com`). The **Domain Name System** 域名系统 (DNS) turns that name into the numeric IP address the browser actually connects to — so people remember names, not numbers.

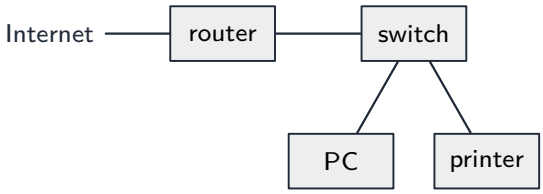


Worked example. You type `example.com`; the DNS returns its IP address (say `93.184.216.34`), and the browser then connects to that number.

■ 3 Wired, wireless and LAN hardware

- **wired** (**Ethernet** 以太网 over **twisted-pair** 双绞线 or **fibre-optic** 光纤 cable): faster, more reliable, more secure.
- **wireless** (Wi-Fi): no cables and devices can move, but slower and easier to intercept.

Key devices: a **network interface card** 网络接口卡 (NIC) connects a device to the network; a **switch** 交换机 sends each message only to the port for its destination; a **router** 路由器 joins a LAN to the internet and forwards data using IP addresses.



Watch out: a switch sends a message only to the *one* device it is meant for — do not confuse it with a router, which connects your whole LAN out to the *internet*.

Practice

2.1 State what an IP address is used for. [1]

2.2 State one difference between the internet and the World Wide Web. [2]

2.3 State what each does: (a) a switch, (b) a router. [2]

2.4 Explain what the DNS does when you type a web address. [2]

2.5 Give one advantage of a wired connection and one advantage of a wireless connection. [2]

2.6 Explain *why* the DNS makes the internet easier for people to use. [2]

2.7 Challenge. A café offers free Wi-Fi to its customers. Give one advantage and one

security risk of using wireless here rather than wired cables. [2]

5 A local area network (LAN) has five computers, one switch and one server.

(a) Describe the characteristics of a LAN.

.....

.....

.....

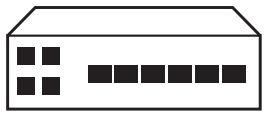
..... [2]

(b) Complete the following diagram to show how these devices are connected in a star topology.

Computer Computer Computer Computer





Computer Server Switch

[2]

(c) Describe how packets are transmitted between two hosts using a star topology.

.....

.....

.....

.....

.....

..... [3]

(d) Another type of network is a bus network.

Ethernet is used to transmit and receive data between the devices on the bus network.

Describe how collisions are detected and managed on this network.

.....

.....

.....

.....

.....

..... [3]

(e) Complete the table by identifying **one** threat to computer and data security posed by networks and the internet.

Describe the threat and give a method of prevention.

Threat	Description	Prevention
.....	.....	.....
.....	.....	.....

[3]

3 Hardware – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
processor	处理器			
primary memory	主存储器			
secondary storage	辅助存储器			
RAM	随机存取存储器			
volatile	易失性			
ROM	只读存储器			
non-volatile	非易失性			
logic gate	逻辑门			
truth table	真值表			

Recall

At IGCSE you named the parts of a computer. Bring this with you:

- A computer has input devices, output devices and storage.
- The **processor** 处理器 (CPU) does the work.

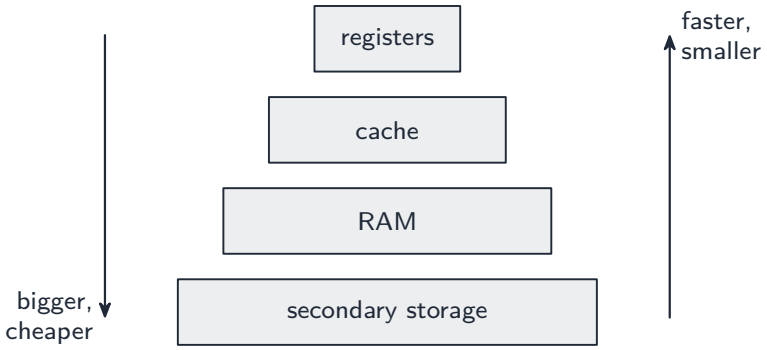
At AS we look more closely at memory and at logic gates — the tiny switches every chip is built from.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Primary and secondary storage

- **primary memory** 主存储器 is the fast memory the processor uses directly while running.
- **secondary storage** 辅助存储器 (hard disk, solid-state drive) keeps data long-term, even when the power is off.



Worked example. When you open a game it is copied from the SSD (secondary) into RAM (primary) so the processor can reach it quickly; closing it frees the RAM, but the game still sits on the SSD.

2 RAM and ROM

- **RAM** 随机存取存储器 holds the programs and data in use; it is **volatile** 易失性 — its contents are lost when the power goes off.
- **ROM** 只读存储器 stores fixed start-up instructions; it is **non-volatile** 非易失性 — kept without power.

RAM
volatile — lost at power off
read *and* write
holds running programs & data

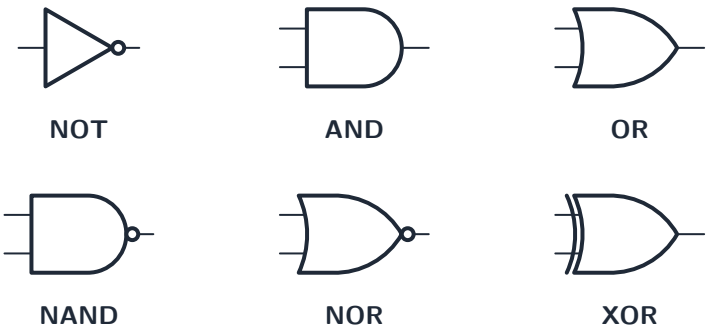
ROM
non-volatile — kept at power off
read-only
holds start-up instructions

Worked example. Switching the computer off erases whatever was in RAM (so unsaved work is lost), but the start-up code in ROM is still there next time you turn it on.

Watch out: "memory" usually means RAM (primary), *not* the hard disk. A file "in memory" is being used now; a file "in storage" is saved for later.

■ 3 Logic gates

A **logic gate** 逻辑门 takes 1s and 0s in and gives a 1 or 0 out. The basic gates are AND, OR and NOT. A **truth table** 真值表 lists the output for every input.



For a 2-input AND gate, the output is 1 only when both inputs are 1:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Worked example. For this AND gate with $A = 1$ and $B = 0$: the inputs are not *both* 1, so the output is 0.

Practice

3.1 Give one example of a secondary storage device. [1]

3.2 A NOT gate has input 1. State its output. [1]

3.3 State one difference between primary memory and secondary storage. [2]

3.4 State whether RAM or ROM is volatile, and what "volatile" means. [2]

3.5 Write the truth table for a 2-input **OR** gate. [2]

3.6 Explain *why* a computer needs both RAM and a hard disk, rather than just one of them. [2]

3.7 Challenge. Write the truth table for the expression `NOT (A AND B)`. [3]

6 **(a)** The processor uses several registers, including the Accumulator (ACC) and the Current Instruction Register (CIR).

Complete the table by describing the role of each register.

Register	Role
ACC	....
CIR	....

[2]

(b) Increasing the number of cores in a processor can affect the performance of a computer.

Describe the drawbacks of increasing the number of cores in a processor.

.....

.....

.....

..... [2]

(c) State **three** differences between Dynamic RAM (DRAM) and Static RAM (SRAM).

1 .....

.....

2 .....

.....

3 .....

.....

[3]

3 Hardware – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
logic circuit	逻辑电路			
Boolean	布尔			
truth table	真值表			
hard disk	硬盘			
solid-state drive	固态硬盘			
optical disc	光盘			

Recall

In **Part 1** you met the computer's parts, RAM and ROM, and the three basic logic gates. Keep these ready:

- A logic gate takes 0s and 1s in and gives a single 0 or 1 out; a truth table lists every case.
- **AND** is 1 only when both inputs are 1; **OR** is 1 when either is 1; **NOT** flips the input.

Part 2 adds three more gates, building them into circuits, and how data is stored long-term.

New ideas

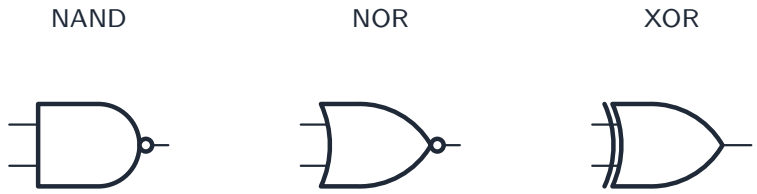
Read these first. Each idea has a short worked example. Then do the Practice.

1 Three more logic gates

Besides AND, OR and NOT there are three more, built from them:

inputs A, B	NAND (not AND)	NOR (not OR)	XOR (different?)
0, 0	1	1	0
0, 1	1	0	1
1, 0	1	0	1
1, 1	0	0	0

NAND and NOR are just AND and OR with the output flipped; **XOR** gives 1 only when the two inputs are **different**.

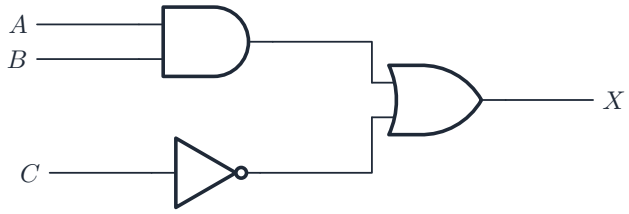


Worked example. XOR is the "are they different?" test: inputs $0, 1 \rightarrow$ output 1 (different); inputs $1, 1 \rightarrow$ output 0 (same).

Watch out: NAND is *not* "NOT then AND" — it is AND with its output flipped (do the AND first, then invert).

2 Logic circuits

Gates joined together make a **logic circuit** 逻辑电路. From a **Boolean** 布尔 expression you can draw the circuit, and from a circuit you can read off the expression and build its **truth table** 真值表.



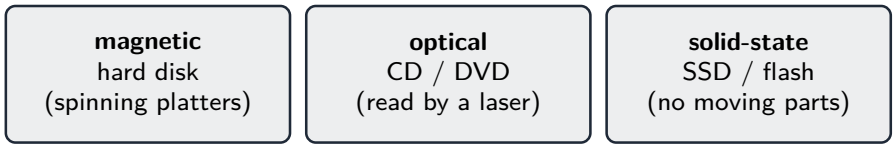
Worked example. For $X = (A$ AND $B)$ OR (NOT C): AND-gate the inputs A, B; NOT-gate C; OR those two results to give X. X is 1 whenever A and B are both 1, **or** C is 0.

3 Storage devices

Secondary storage keeps data when the power is off. The main types:

- a **hard disk** 硬盘 (HDD): data on spinning magnetic disks (cheap, large, has moving parts);
- a **solid-state drive** 固态硬盘 (SSD): flash memory, no moving parts (faster, more robust);

- an **optical disc** 光盘 (CD / DVD): tiny pits read by a laser.



Worked example. A laptop often uses an SSD as its main drive (fast, silent, no moving parts) and a large external hard disk for backups (cheaper per gigabyte).

Practice

3.1 Which storage device is read using a laser? [1]

3.2 State one advantage of an SSD over a hard disk. [1]

3.3 Complete the output column of this truth table for a 2-input **NAND** gate. [2]

A	B	A NAND B
0	0	
0	1	
1	0	
1	1	

3.4 State the input combinations (of two inputs) for which an **XOR** gate outputs 0, and those for which it outputs 1. [2]

3.5 A circuit ANDs inputs A and B, then ORs that result with input C. Write the Boolean expression for the output X. [2]

3.6 Explain *why* a NAND gate always gives the opposite output to an AND gate. [2]

3.7 Challenge. Using only AND, OR and NOT, write a Boolean expression that is 1 only when A and B are *different* (i.e. build an XOR). [2]

3 **(a)** Complete the truth table for the following logic expression:

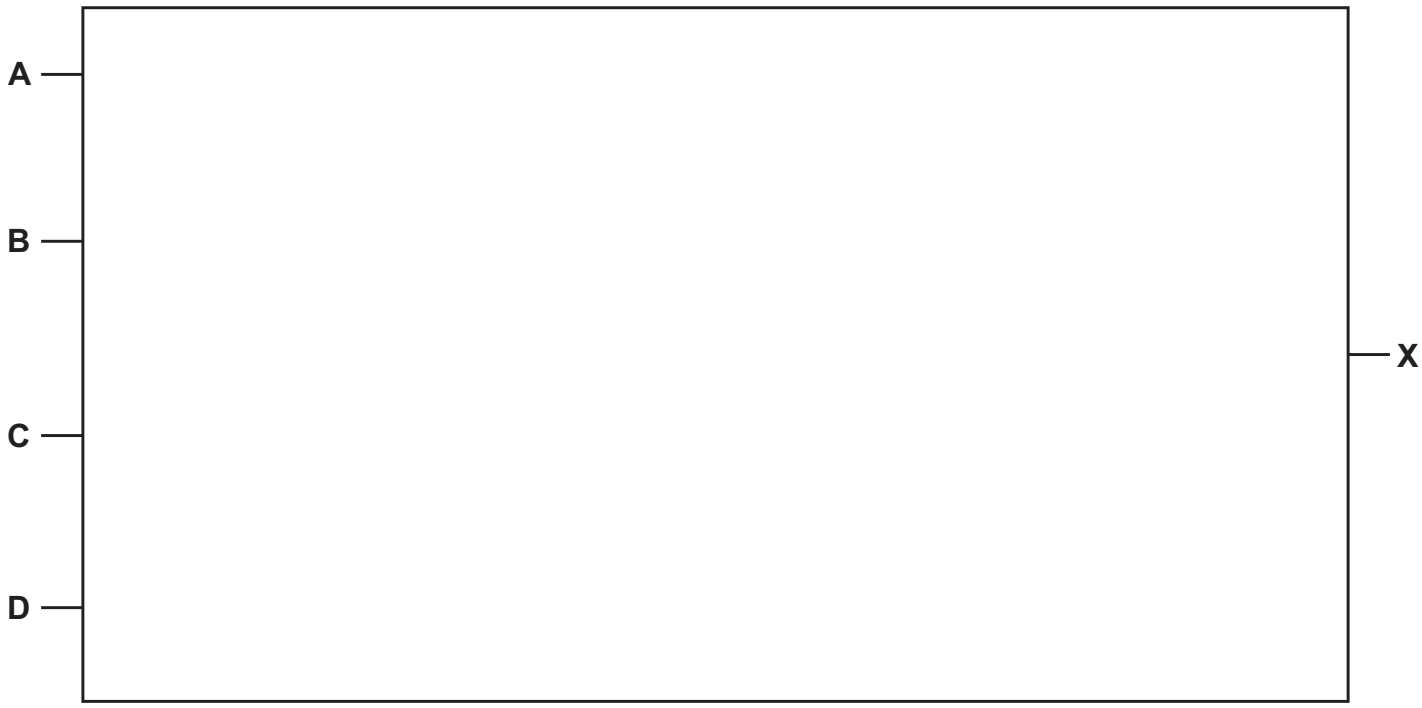
X = (**A** NOR **B**) NAND (**C** XOR **B**)

A	**B**	**C**	**Working space**	**X**
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[2]

(b) Draw a logic circuit for the logic expression:

X = NOT (((**A** AND **B**) OR **C**) AND (**C** NOR **D**))



[2]

4 Processor Fundamentals – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
processor	处理器			
stored program	存储程序			
arithmetic and logic unit	算术逻辑单元			
control unit	控制单元			
register	寄存器			
Program Counter	程序计数器			
Accumulator	累加器			
memory address register	存储器地址寄存器			
memory data register	存储器数据寄存器			
current instruction register	当前指令寄存器			
fetch–execute cycle	取指执行周期			
data bus	数据总线			
address bus	地址总线			

Recall

At IGCSE you know the CPU runs programs. Bring this with you:

- The **processor** 处理器 fetches and carries out instructions.
- Programs and data are stored in memory.

At AS we look inside the processor and at how it runs a single instruction — the cycle every computer repeats billions of times a second.

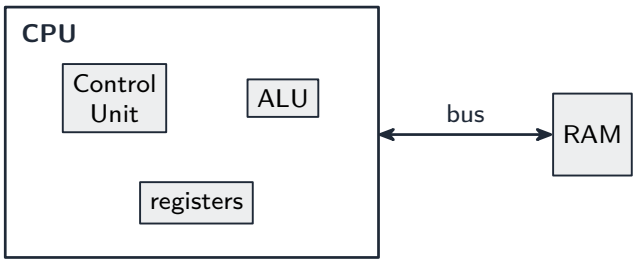
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Inside the processor

A computer keeps its instructions and data together in memory — the **stored program** 存储程序 idea. The processor contains:

- an **arithmetic and logic unit** 算术逻辑单元 that does the maths and logic;
- a **control unit** 控制单元 that directs everything;
- small, very fast stores called **register** 寄存器 cells.



Worked example. To work out $3 + 4$: the control unit fetches and decodes the "add" instruction, the ALU adds the two numbers, and a register holds the answer 7.

2 Two important registers

- the **Program Counter** 程序计数器 holds the address of the next instruction to run;
- the **Accumulator** 累加器 holds the result of a calculation.

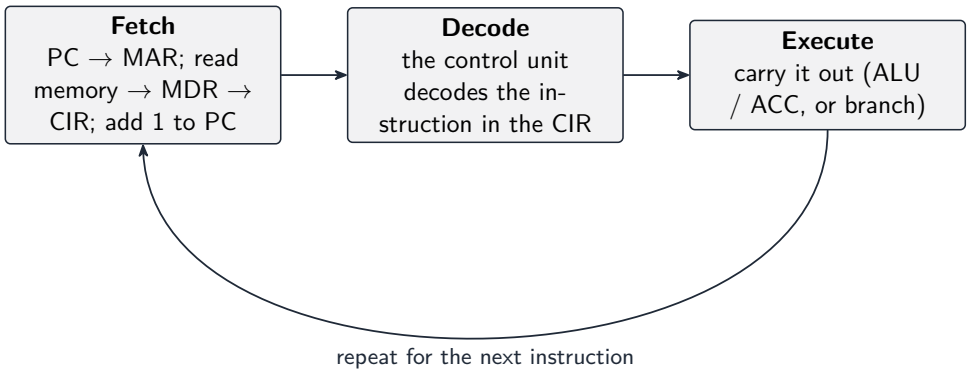
Worked example. Say the Program Counter holds 30. The CPU copies 30 into the **memory address register** 存储器地址寄存器 (MAR) and reads memory location 30; the instruction arrives in the **memory data register** 存储器数据寄存器 (MDR), then moves to the **current instruction register** 当前指令寄存器 (CIR), ready to be run.

Watch out: the Program Counter holds the *address* of the next instruction, not the instruction itself.

■ 3 The fetch–execute cycle

To run a program the processor repeats the **fetch–execute cycle** 取指执行周期:

1. **fetch** the next instruction from memory,
2. **decode** it (work out what it means),
3. **execute** it (carry it out).



Inside, data travels on the **data bus** 数据总线 and memory addresses travel on the **address bus** 地址总线.

Worked example. For "add the number at address 50": *fetch* the instruction from memory, *decode* it as an add, then *execute* it — the ALU adds the value stored at address 50 into the accumulator.

Practice

4.1 State what the Program Counter holds. [1]

4.2 State what travels on the address bus. [1]

4.3 Name the two main units inside the processor and say what each does. [2]

4.4 State, in order, the three steps of the fetch–execute cycle. [3]

4.5 Explain *why* the Program Counter must be increased during each fetch–execute cycle. [2]

4.6 Challenge. The Program Counter holds 30. Describe how the registers are used to fetch the instruction stored there. [3]

8 A computer designed using the Von Neumann model for a computer system contains general purpose registers and special purpose registers.

(a) Describe the purpose of the Status Register (SR).

.....

.....

.....

..... [2]

(b) Identify **two** differences between general purpose registers and special purpose registers.

1 .....

.....

.....

2 .....

.....

.....

[2]

4 Processor Fundamentals – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
machine code	机器码			
Assembly language	汇编语言			
mnemonics	助记符			
assembler	汇编器			
opcode	操作码			
operand	操作数			
addressing mode	寻址方式			
immediate addressing	立即寻址			
direct addressing	直接寻址			
indexed addressing	变址寻址			
logical shift	逻辑移位			
mask	掩码			

Recall

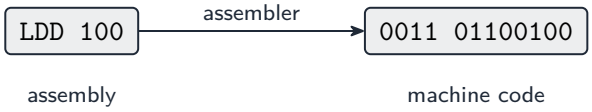
In **Part 1** you met the Von Neumann CPU (ALU, control unit, registers like the PC and accumulator), the fetch–execute cycle, and the three buses. Part 2 looks at the language the CPU runs, how it finds its data, and working on single bits — the level a real processor actually works at.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Machine code and assembly language

The CPU actually runs **machine code** 机器码 — bit patterns. **Assembly language** 汇编语言 is a readable form with one instruction each, written using short **mnemonics** 助记符 like `LDD`, `ADD`, `JMP`. An **assembler** 汇编器 translates the assembly into machine code.

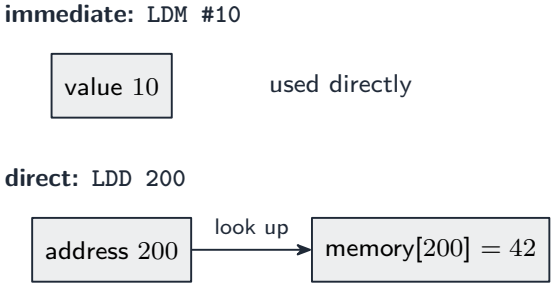


Worked example. The assembler turns the readable `ADD 100` into a pattern of bits: an **opcode** 操作码 part that means "ADD", and an **operand** 操作数 part that holds 100.

2 Addressing modes

The **addressing mode** 寻址方式 says how the CPU finds the operand of an instruction:

- **immediate addressing** 立即寻址 — the value is *in* the instruction (`LDM #10` loads the number 10);
- **direct addressing** 直接寻址 — the instruction holds an *address*; the operand is the value stored there (`LDD 200`);
- **indexed addressing** 变址寻址 — the address is a base plus an index register, used to step through an array.



Worked example. Suppose memory location 10 holds the value 99. Then `LDM #10`

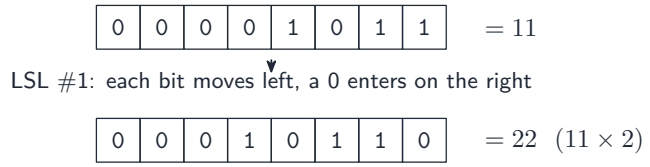
loads 10 (the number written in the instruction), but `LDD 10` loads 99 (the value found at address 10) — same number, very different result.

Watch out: the `#` changes everything. `LDM #10` uses the number 10; `LDD 10` uses whatever is *stored at* address 10.

■ 3 Binary shifts and bit masks

A **logical shift** 逻辑移位 moves all the bits left or right, filling the empty places with 0:

- left by 1 → multiply by 2; right by 1 → integer divide by 2.



A **mask** 掩码 lets you change one chosen bit: **OR** with a mask **sets** a bit, **AND** with a mask **clears** a bit, **XOR** with a mask **toggles** a bit.

Worked example. To set bit 0 of `01000000` without touching the rest, OR it with the mask `00000001`: the result is `01000001`.

Practice

4.1 State what an assembler does. [1]

4.2 State which bitwise operation (AND, OR or XOR with a mask) is used to **set** a chosen bit to 1. [1]

4.3 State what each instruction loads: (a) `LDM #10`, (b) `LDD 200`. [2]

4.4 Apply `LSL #1` (left shift by one) to `00001011`. Give the result and the arithmetic operation it performs. [2]

4.5 Apply `LSR #1` (right shift by one) to `00011000`. Give the result and the operation it performs. [2]

4.6 Explain *why* a logical left shift by one is the same as multiplying by 2. [2]

4.7 Challenge. Show how to multiply 5 (`00000101`) by 4 using a logical shift. Give the shift used and the final byte. [2]

11 The table shows assembly language instructions for a processor that has one register, the Accumulator (ACC).

Label	Instruction		Explanation
	Opcode	**Operand**	
	LDM	#n	Load the number n to ACC
	LDD	<address>	Load the contents of the location at the given address to the ACC
	LDI	<address>	The address to be used is at the given address. Load the contents of this second address to the ACC
	ADD	<address>	Add the contents of the given address to the ACC
	SUB	<address>	Subtract the contents of the given address from the ACC
	STO	<address>	Store the contents of the ACC at the given address
<label>:		<data>	Gives a symbolic address <label> to the memory location with contents <data>
# denotes a denary number, e.g. #123 <label> can be used in place of <address>			

(a) Write **assembly language** code, using **only** the given instruction set to:

- store the denary value 100 as a named constant
- subtract the constant from the value contained in address 632
- store the result in variable Answer.

Show the initialisation of the constant and Answer in the table provided.

.....

.....

.....

.....

.....

.....

Label	Contents

[6]

(b) The address 632 contains the value 45.

State the value of Answer after the code described in part **(a)** has executed.

.....

[1]

8 **(a)** Convert the hexadecimal number 1FAB into denary.

Working .....

.....

.....

Denary value ..... [1]

(b) Explain how to convert the two's complement integer 10011111 into denary. Give the denary value after conversion.

Explanation .....

.....

.....

.....

.....

Denary value .....
[3]

(c) Describe the difference between a right logical binary shift and a right arithmetic binary shift.

.....

.....

.....

..... [2]

5 System Software – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
operating system	操作系统			
process	进程			
virtual memory	虚拟内存			
device driver	设备驱动			
utility program	实用程序			
translator	翻译器			
compiler	编译器			
interpreter	解释器			
assembler	汇编器			

Recall

At IGCSE you used computers every day. Bring this with you:

- The operating system is the main software that runs the machine.
- You ran programs (applications) on top of it.

At AS we list what the operating system does, and meet the programs that translate code — the software that makes all other software possible.

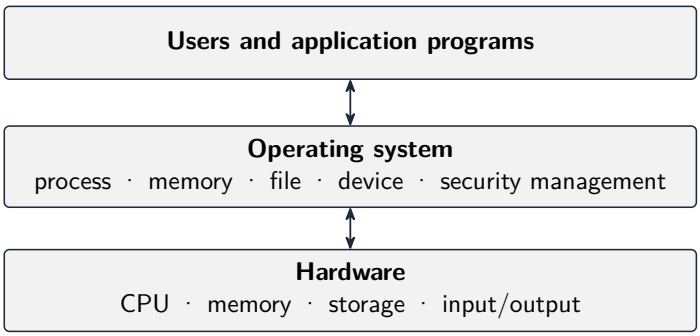
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 The operating system

The **operating system** 操作系统 manages the computer's hardware and software so that programs can run. Its jobs include:

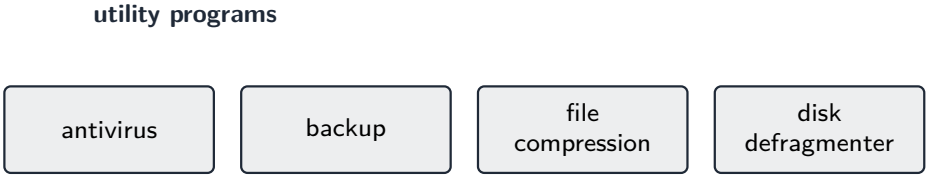
- managing each running program (a **process** 进程);
- managing memory, including **virtual memory** 虚拟内存 (using disk space as extra RAM when memory is full);
- talking to each piece of hardware through a **device driver** 设备驱动.



Worked example. When you open a photo editor, the OS gives it memory, loads it from disk, and lets it use the screen and mouse through device drivers — the program never has to know the hardware details itself.

2 Utility programs

A **utility program** 实用程序 is a small tool that keeps the system healthy — antivirus, backup and file compression are common examples.

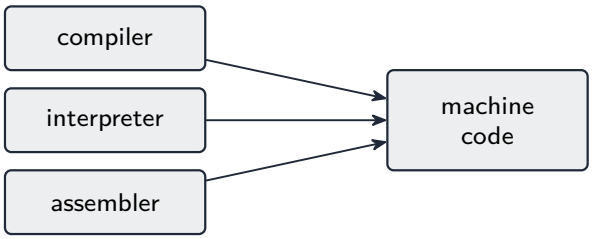


Worked example. A backup utility copies your files to another disk on a schedule, so a crash or theft does not lose your work.

3 Translators

Code you write must be turned into machine code by a **translator** 翻译器:

- a **compiler** 编译器 translates the whole program at once, then it can run on its own;
- an **interpreter** 解释器 translates and runs one line at a time;
- an **assembler** 汇编器 turns assembly language into machine code.



Worked example. Python usually uses an interpreter (runs line by line, easy to test as you go); C uses a compiler (translated once into a fast program that runs on its own).

Watch out: an interpreter does *not* leave behind a machine-code file you can keep — it translates the code again every time the program runs.

Practice

5.1 Give one example of a utility program. [1]

5.2 State what an assembler translates, and into what. [1]

5.3 State two jobs done by the operating system. [2]

5.4 Explain what virtual memory is. [2]

5.5 State one difference between a compiler and an interpreter. [2]

5.6 Explain one advantage of using an interpreter while you are still writing and testing a program. [2]

5.7 Challenge. A finished program will be sent to customers who must not see the source code and who want it to run fast. Would you give them a compiled or an interpreted version? Give a reason. [2]

6 A computer has an Operating System (OS).

Memory management and process management are two OS tasks.

Explain how memory management **and** process management support multi-tasking.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

5 System Software – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
compiler	编译器			
executable	可执行文件			
interpreter	解释器			
bytecode	字节码			
virtual machine	虚拟机			
integrated development environment	集成开发环境			
debugger	调试器			
breakpoints	断点			

Recall

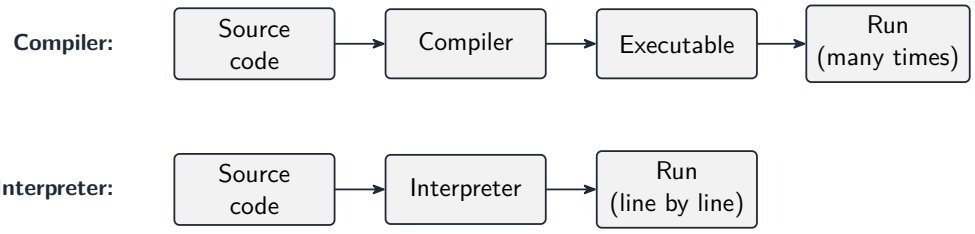
In **Part 1** you met the operating system, utility programs, and the three translators (assembler, compiler, interpreter). Part 2 compares compilers and interpreters more closely, meets the clever "bytecode + virtual machine" idea, and looks at the IDE programmers work in.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Compiler vs interpreter

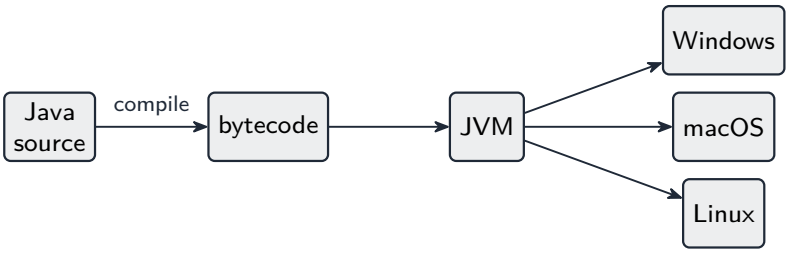
- A **compiler** 编译器 translates the **whole** program once into a standalone **executable** 可执行文件. It then runs fast and needs no translator present — but it is tied to one type of computer, and it reports all the errors together at the end.
- An **interpreter** 解释器 translates and runs **one line at a time**. It is slower and must stay installed, but it is easy to edit-and-rerun and to move between computers, and it stops at the first faulty line.



Worked example. A game written in C++ is compiled once and shipped as one fast program; a small Python script is interpreted, so you can change a line and rerun it at once.

2 Bytecode and the virtual machine

Java is compiled into **bytecode** 字节码 — a platform-independent in-between form. A **virtual machine** 虚拟机 (the JVM) on any computer then runs that bytecode. This is "write once, run anywhere": one compiled file runs on Windows, macOS or Linux.

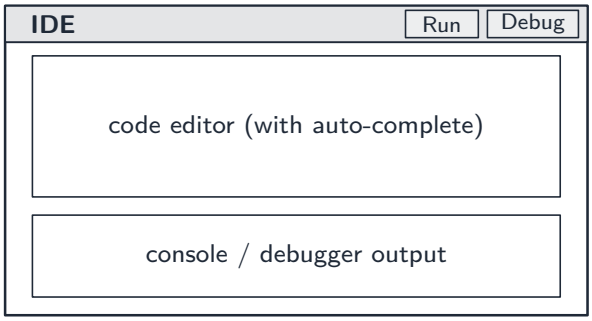


Worked example. A Java app compiled to bytecode on a Windows laptop runs unchanged on a Mac or a phone, because each has its own virtual machine to run that same bytecode.

Watch out: bytecode is *not* machine code — it still needs a virtual machine to run it. That extra step is exactly what lets it run anywhere.

3 The IDE

An **integrated development environment** 集成开发环境 (IDE) puts all the tools in one program: a code editor with colour **syntax highlighting** and auto-complete, one-key compile-and-run, and a **debugger** 调试器. With the debugger you set **breakpoints** 断点 to pause the program and inspect the variables, or step through it one line at a time.



Worked example. A breakpoint set on line 12 pauses the program there; you see that `total` is 0 when you expected 50, which points you straight to the bug.

Practice

5.1 State whether a compiled program needs the compiler installed in order to run. [1]

5.2 State what a breakpoint does in a debugger. [1]

5.3 State one difference between a compiler and an interpreter. [2]

5.4 Name two features an IDE provides to help write code. [2]

5.5 Explain why a Java program compiled to bytecode can run on many different computers. [2]

5.6 Explain one reason a debugger with breakpoints can find a bug faster than adding print statements all over the code. [2]

5.7 Challenge. A company wants one app to run on Windows, Mac and Linux without rewriting it. Explain how compiling to bytecode and using a virtual machine makes this possible. [2]

3 A software developer is writing a computer program.

(a) The developer uses an interpreter while writing the program code because it is easier for debugging.

Explain **one** reason why it is easier to debug the program code using an interpreter instead of a compiler.

.....

.....

.....

..... [2]

(b) The program is ready to be sold to customers.

The developer uses a compiler because it creates an executable file.

Explain the reasons why the need to create an executable file makes the complier the appropriate choice when the program is complete.

.....

.....

.....

.....

.....

..... [3]

6 Security, privacy and data integrity – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
secure	安全			
malware	恶意软件			
phishing	网络钓鱼			
encryption	加密			
firewall	防火墙			
authentication	身份验证			
validation	验证			
verification	核对			
check digit	校验位			
parity check	奇偶校验			

Recall

At IGCSE you met online dangers and passwords. Bring this with you:

- Data needs to be kept **secure** 安全 from people who should not see or change it.
- You used passwords and antivirus software.

At AS we name the main threats, the ways to protect data, and how to check data is correct.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

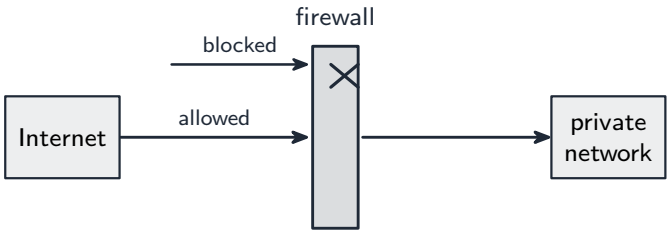
1 Threats

- **malware** 恶意软件 is harmful software — viruses, worms and Trojans.
- **phishing** 网络钓鱼 tricks people into giving away passwords using fake messages or websites.

Worked example. A phishing email pretends to be your bank and links to a fake login page; if you type your password there, the attacker captures it.

2 Protecting data

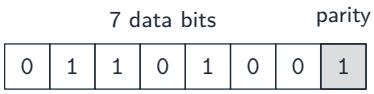
- **encryption** 加密 scrambles data so that only someone with the correct key can read it.
- a **firewall** 防火墙 controls which traffic may enter or leave a network.
- **authentication** 身份验证 checks who you are — a password, a fingerprint, or two-factor codes.



Worked example. Encryption turns a message like `HELLO` into scrambled text such as `XZ4PQ`; intercepted without the key, it means nothing.

3 Keeping data correct (integrity)

- **validation** 验证 checks that entered data is sensible (for example a range check or a type check).
- **verification** 核对 checks that data was copied or typed correctly (for example double entry).
- a **check digit** 校验位 is an extra digit added to a number to catch typing errors; a **parity check** 奇偶校验 adds an extra bit so a single flipped bit is noticed.



data has three 1s; for **even parity** the parity bit is **1** (four 1s in total)

Worked example. For even parity, count the 1s in `0110100` — that is three (odd) — so the parity bit is `1`, which makes the total even.

Watch out: validation only checks data is *sensible*, not *true* — a valid-looking but wrong phone number can still pass a length check.

Practice

6.1 Give one method of authentication. [1]

6.2 Explain what encryption does. [1]

6.3 State what malware is, and give one example. [2]

6.4 State one difference between validation and verification. [2]

6.5 A system uses **even** parity. The seven data bits are 1010101. State the parity bit

needed. [2]

6.6 Explain *why* a parity check can fail to notice an error if *two* bits are flipped. [2]

6.7 Challenge. A website wants to protect its users' passwords. Suggest two different measures and state what each one protects against. [2]

8 One ethical consideration for a student connecting their personal computer to the school network is the risk of spreading malware on the network.

(a) Viruses and pharming are examples of malware.

Explain what is meant by a virus and pharming.

Virus .....

.....

.....

Pharming .....

.....

.....

[2]

(b) Give **three** other ethical considerations for a student using their personal computer to connect to the school network.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(c) Describe **two** social impacts of students using Artificial Intelligence (AI) to complete their homework.

1 .....

.....

.....

.....

2 .....

.....

.....

....

[4]

6 Security, privacy and data integrity – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
plaintext	明文			
ciphertext	密文			
symmetric encryption	对称加密			
asymmetric encryption	非对称加密			
public key	公钥			
private key	私钥			
digital signature	数字签名			
denial of service	拒绝服务			
man-in-the-middle	中间人攻击			
checksum	校验和			

Recall

In **Part 1** you met threats (malware, phishing), protections (firewall, encryption, authentication), and data integrity (validation, verification, the check digit and parity). Part 2 looks at how encryption really works, two more network attacks, and stronger transfer checks.

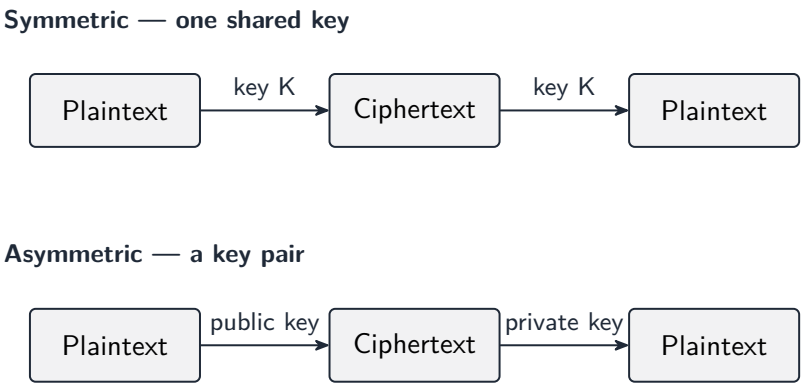
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Symmetric vs asymmetric encryption

Encryption turns readable **plaintext** 明文 into scrambled **ciphertext** 密文 using a key.

- **symmetric encryption** 对称加密 uses **one shared key** to both lock and unlock — fast, but the key itself must be shared safely.
- **asymmetric encryption** 非对称加密 uses a pair: a **public key** 公钥 to lock (which anyone may have) and a **private key** 私钥 to unlock (kept secret). This solves the "how do we share the key" problem.



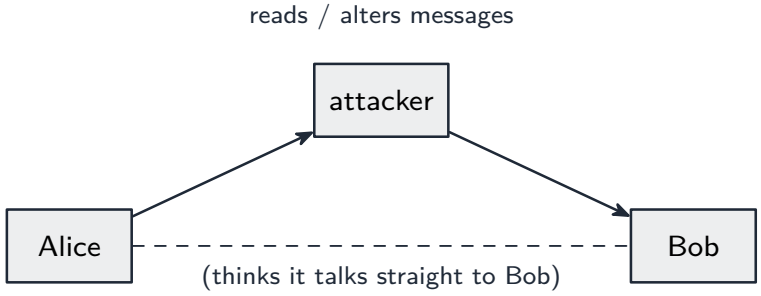
Worked example. To receive secret messages, Bob publishes his public key; anyone can lock a message with it, but only Bob's private key can unlock it — so no secret key is ever shared.

2 Digital signatures and two more attacks

A **digital signature** 数字签名 proves *who* sent a message and that it was *not changed* on the way.

Two network attacks to know:

- a **denial of service** 拒绝服务 (DoS) attack floods a server with traffic so real users cannot get through;
- a **man-in-the-middle** 中间人攻击 attacker secretly sits between two parties, reading or altering their messages.



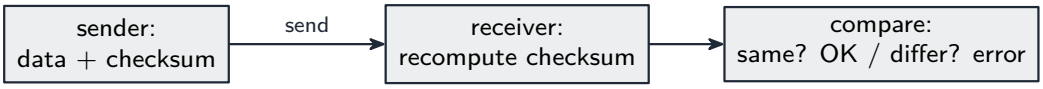
Worked example. On public Wi-Fi, a man-in-the-middle relays your messages to the bank while quietly reading them — both sides think they are talking directly.

Watch out: in asymmetric encryption you lock with the *recipient's public* key, not your own — only *their* private key can then unlock it.

■ 3 Checking a transfer arrived correctly

When data is sent, bits can flip. The receiver checks it:

- a parity bit catches a single flipped bit (Part 1);
- a **checksum** 校验和 is a summary number sent with the data; the receiver recomputes it and compares — a mismatch means an error. (A cyclic redundancy check, CRC, is a stronger version.)



Worked example. The sender adds the byte values to get a checksum of 204 and sends it too; the receiver re-adds the bytes — 204 means the data is intact, anything else means an error.

Practice

6.1 In asymmetric encryption, state which key is used to **encrypt** a message sent to someone. [1]

6.2 State one difference between symmetric and asymmetric encryption. [2]

6.3 State what a digital signature proves about a message. [2]

6.4 Describe what a denial-of-service (DoS) attack does. [2]

6.5 Explain how a checksum lets the receiver detect an error in transferred data. [2]

6.6 Explain *why* asymmetric encryption solves a problem that symmetric encryption has. [2]

6.7 Challenge. A simple checksum just adds up the byte values. Explain why it might miss an error where two bytes are *swapped*. [2]

1 Draw **one** line from each verification method to indicate whether it is used during data transfer or data entry.

Verification Method	Use
Parity byte check	
Checksum	Data transfer
Visual check	
Parity block check	Data entry

[2]

7 Ethics and Ownership – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
Ethics	伦理			
privacy	隐私			
Intellectual property	知识产权			
Copyright	版权			
patent	专利			
software licence	软件许可证			
open-source	开源			
freeware	免费软件			
shareware	共享软件			
Artificial intelligence	人工智能			
bias	偏见			

Recall

Computers raise questions of right and wrong, and of who owns software and data. You have used apps, games and websites made by other people.

At AS we look at ethics, at owning software (licences), and a little about artificial intelligence.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 Ethics and privacy

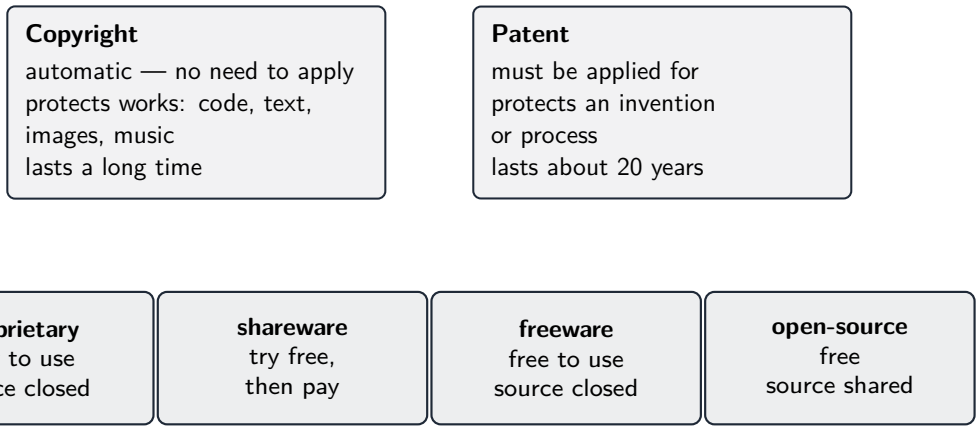
Ethics 伦理 means acting responsibly with computers — protecting people's **privacy** 隐私, keeping data safe, and not using it in ways people did not agree to.

Worked example. A fitness app that quietly sells users' location data to advertisers breaks their privacy — even if the app itself works perfectly.

■ 2 Owning software and ideas

Intellectual property 知识产权 is something a person created with their mind — a program, a design, a song.

- **Copyright** 版权 protects a created work (it applies automatically).
- A **patent** 专利 protects a new invention.
- A **software licence** 软件许可证 sets out how you may use a program:
  - **open-source** 开源 — the source code is shared and may be changed and re-shared;
  - **freeware** 免费软件 — free to use, but the source code is not shared;
  - **shareware** 共享软件 — free to try for a limited time, then you pay.

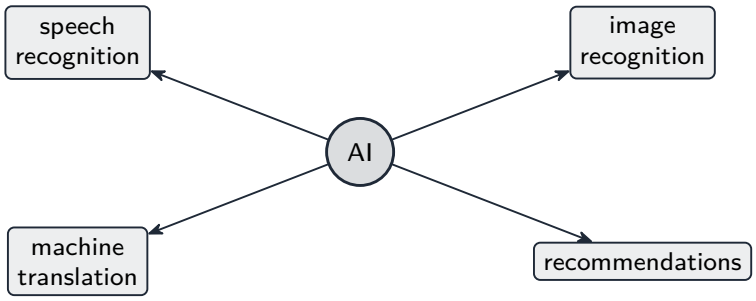


Worked example. With an open-source program you may read its code, change it, and share your version; with freeware you may use it for free but cannot see or change the code.

Watch out: "free to use" (freeware) is not the same as "open-source" — freeware is still free, but it keeps its source code hidden.

■ 3 Artificial intelligence

Artificial intelligence 人工智能 lets a computer learn patterns from data and make decisions. It is powerful and useful, but it can show **bias** 偏见 if the data it learns from is unfair or incomplete.



Worked example. A face-recognition system trained mostly on one group of faces may work poorly on others — that unfair result is bias coming from the training data.

Practice

7.1 State what is meant by intellectual property. [1]

7.2 Explain what is meant by bias in an AI system. [1]

7.3 State what each of copyright and a patent protects. [2]

7.4 Give one difference between open-source software and freeware. [2]

7.5 Give one ethical concern about a company collecting people's personal data. [2]

7.6 A student copies an open-source library into their project but removes the author's name and licence. Explain why this is wrong, even though the code was free to use. [2]

7.7 Challenge. A hospital wants to use AI to decide which patients to treat first. Give one benefit and one ethical risk. [2]

7 Ethics and Ownership – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
Artificial intelligence	人工智能			
machine learning	机器学习			
neural networks	神经网络			
speech recognition	语音识别			
image recognition	图像识别			
machine translation	机器翻译			
optical character recognition	光学字符识别			
text-to-speech	文本转语音			
bias	偏见			

Recall

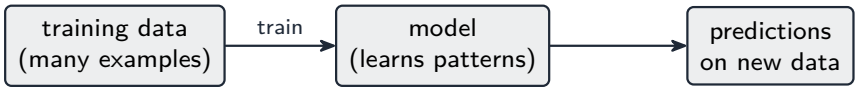
In **Part 1** you met ethics, privacy, intellectual property (copyright and patents), the kinds of software licence, and artificial intelligence with its risk of bias. Part 2 goes deeper into AI — how it learns, what it is used for, and its benefits and dangers.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 How modern AI learns

Artificial intelligence 人工智能 builds systems that do tasks once thought to need human intelligence. Most modern AI uses **machine learning** 机器学习 — the system **learns patterns from large amounts of data** instead of being programmed step by step. The leading method today is deep learning, built on **neural networks** 神经网络 (many connected layers, loosely inspired by the brain).

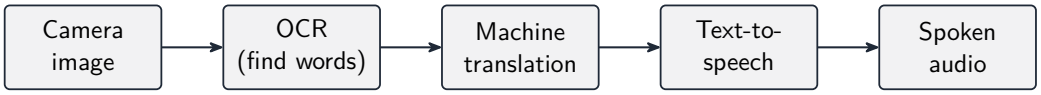


Worked example. To recognise cats, a model is shown thousands of labelled photos and learns the patterns itself — nobody writes a rule like "a cat has pointed ears".

2 What AI is used for

- *Understanding input:* **speech recognition** 语音识别 (turning speech into text) and **image recognition** 图像识别 (finding objects or faces in a picture).
- *Producing output or decisions:* **machine translation** 机器翻译, product recommendations, and self-driving cars.

A common chain: a phone reads a foreign label — **optical character recognition** 光学字符识别 finds the words, machine translation converts them, and **text-to-speech** 文本转语音 reads the result aloud.



Worked example. A "recommended for you" list comes from a model that learned which items people with similar tastes also liked.

3 Benefits and concerns

- **Benefits:** accessibility (helping people with impairments), productivity (automating dull tasks), spotting patterns in huge datasets, and being always available.
- **Concerns: bias** 偏见 (unfair training data → unfair decisions), job loss, privacy (training uses personal data), "black-box" decisions that are hard to explain, and accountability when AI gets it wrong.

Worked example. A bank's AI trained on past loans may reject a fair applicant because the old data was biased — a fast decision, but an unfair one.

> **Watch out:** an AI is only as fair as its training data — "the computer decided" does not make a decision correct or unbiased.

Practice

7.1 Give one example of an AI task that *understands input* and one that *produces an output*. [2]

7.2 Explain what "machine learning" means, and how it differs from ordinary step-by-step programming. [2]

7.3 A phone reads a foreign sign aloud. Name the three AI steps it uses, in order. [3]

7.4 Explain what is meant by bias in an AI system, and where it comes from. [2]

7.5 Give one benefit and one concern of using AI. [2]

7.6 Explain *why* a "black-box" AI decision (one that cannot be explained) is a problem when the AI makes an important choice. [2]

7.7 Challenge. A company replaces its customer-service staff with an AI chatbot. Give one benefit to the company and one risk to its customers. [2]

7 Software is distributed with a licence.

(a) Give **two** benefits of distributing software using a shareware software licence.

1 .....

.....

2 .....

.....

[2]

(b) Give **two** benefits of distributing software using a commercial software licence.

1 .....

.....

2 .....

.....

[2]

8 Databases – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
relational database	关系数据库			
table	表			
record	记录			
field	字段			
primary key	主键			
foreign key	外键			
referential integrity	参照完整性			
normalisation	规范化			
query	查询			
DBMS	数据库管理系统			

Recall

At IGCSE you stored data in a single table of rows and columns. Bring this with you:

- A table holds data in rows (one per item) and columns (one per piece of information).

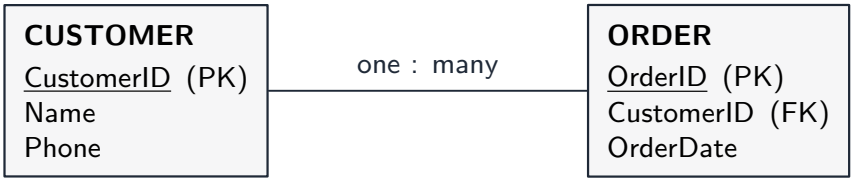
At AS we link several tables together (a relational database) and use a language to ask it questions.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Why a relational database

Putting everything in one big flat file repeats the same data many times (wasteful and easy to get wrong). A **relational database** 关系数据库 splits the data into linked **table** 表 units. In a table, each row is a **record** 记录 and each column is a **field** 字段.

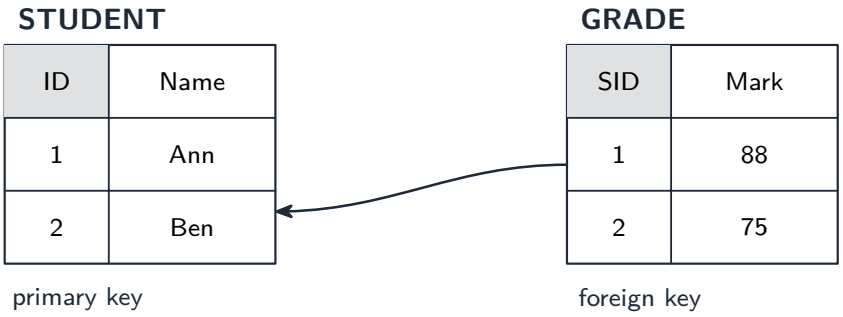


The **foreign key** CustomerID in ORDER refers to the **primary key** CustomerID in CUSTOMER, linking each order to its customer.

Worked example. If every borrowing record stored a member's full address, changing one address would mean editing dozens of rows; keeping members in their own table means you change it once.

2 Keys

- the **primary key** 主键 is the field that gives each record a unique label;
- a **foreign key** 外键 is a primary key from another table, used to link the two tables;
- **referential integrity** 参照完整性 means every foreign key must point to a record that really exists.



Splitting the data neatly so nothing is repeated is called **normalisation** 规范化.

Worked example. In a `Loans` table, the `MemberID` foreign key matches the `MemberID` primary key in the `Members` table, so each loan links to exactly one member.

■ 3 Asking questions with SQL

A **query** 查询 asks the database for chosen data. The language is SQL, and a **DBMS** 数据库管理系统 (database management system) runs it. For example:

```
SELECT name, grade
FROM   students
WHERE  grade > 80;
```

Worked example. That query returns the `name` of only those students whose `grade` is above 80 — the `WHERE` line filters out the rest.

Watch out: `SELECT *` returns *every* column. List just the fields you need (e.g. `SELECT name, grade`) when you don't want them all.

Practice

8.1 State one problem with storing all data in a single flat file. [1]

8.2 Explain what a primary key is. [1]

8.3 State the difference between a record and a field. [2]

8.4 Explain what a foreign key is used for. [2]

8.5 Write a SQL query that selects **all** fields from a table called `Books`. [2]

8.6 Explain *why* splitting data into linked tables makes updating it safer than one big flat file. [2]

8.7 Challenge. Write a SQL query that selects the `title` and `author` of books from a `Books` table where the `year` is after 2000. [2]

4 A relational database, SHIPPING, stores data about the ships in a company and the containers that are carried on the ships.

The database has the following tables:

CONTAINER(ContainerID, Type, Weight, OwnerName, ShipID)

SHIP(ShipID, Type, Capacity, ShipName)

(a) Describe the relationship between the two tables. Refer to the primary and foreign keys in your answer.

.....

.....

.....

..... [2]

(b) The table CONTAINER needs an additional field to store the data for the last inspection date.

Write a Structured Query Language (SQL) script to add **one** field to the table CONTAINER to store the date of last inspection of the container, for example 08/07/2019.

.....

.....

.....

..... [2]

(c) Write an SQL script to return the number of containers stored in the database for the ship with the name Caledonia.

.....

.....

.....

.....

.....

..... [4]

(d) Describe the purpose of a developer interface in a Database Management System (DBMS).

.....

.....

.....

..... [2]

8 Databases – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
entity	实体			
entity-relationship diagram	实体关系图			
cardinality	基数			
link table	连接表			
normal forms	范式			
Data Definition Language	数据定义语言			
Data Manipulation Language	数据操纵语言			
join	连接			
aggregate functions	聚合函数			

Recall

In **Part 1** you met the relational model (tables, records, fields, primary and foreign keys), referential integrity, and a simple `SELECT` query. Part 2 adds diagramming a database, tidying it up, and more SQL.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 Entity-relationship (E-R) diagrams

An **entity** 实体 is a thing we store data about (a `CUSTOMER`). An **entity-relationship diagram** 实体关系图 draws each entity as a box joined by relationship lines, marked with the **cardinality** 基数:

- *one-to-one*, *one-to-many* (one customer has many orders), *many-to-many*.

A many-to-many relationship can't be stored directly — you add a **link table** 连接表 holding the two foreign keys (e.g. an `ENROLMENT` table between `STUDENT` and `COURSE`).

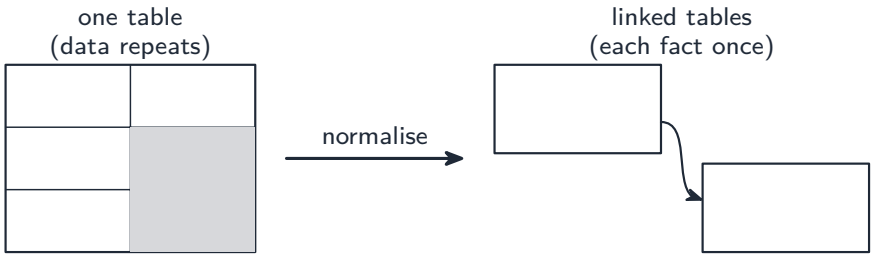
Worked example. `STUDENT` and `COURSE` are many-to-many (a student takes many courses; a course has many students), so an `ENROLMENT` link table holds both keys.

■ 2 Normalisation

Normalisation organises tables into **normal forms** 范式 to remove repeated data:

- **1NF** — every field holds a single value (no repeating groups), with a primary key;
- **2NF** — no field depends on only *part* of a composite key;
- **3NF** — no non-key field depends on another non-key field.

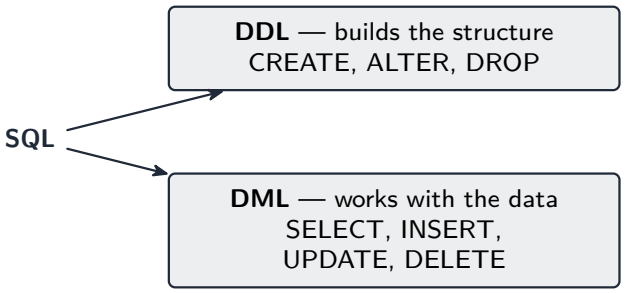
Aim for **3NF**: each fact is stored once, so updates can't make copies disagree.



Worked example. A table that repeats a customer's name beside each of their orders is moved towards 3NF by holding the name once in a `CUSTOMER` table and linking by key.

■ 3 SQL: DDL and DML

SQL has two halves: **Data Definition Language** 数据定义语言 (DDL) builds the *structure*, and **Data Manipulation Language** 数据操纵语言 (DML) works with the *data*. A **join** 连接 combines two tables on a key; **aggregate functions** 聚合函数 (`COUNT`, `SUM`, `AVG`) summarise rows.




```
SELECT Name, Phone
FROM   CUSTOMER
WHERE  City = 'London'
ORDER BY Name;
```

Worked example. `CREATE TABLE` (DDL) builds an empty table; `INSERT INTO` (DML) then adds rows of data into it.

Watch out: a `JOIN` needs a key that appears in *both* tables — with no shared key field you cannot link them.

Practice

8.1 A customer can place many orders, but each order belongs to one customer. State the cardinality of this relationship. [1]

8.2 State whether `CREATE TABLE` is a DDL or a DML statement. [1]

8.3 State how a many-to-many relationship is stored in a relational database. [1]

8.4 State what it means for a table to be in first normal form (1NF). [2]

8.5 Write a SQL query that selects the `Name` of every row in a table `STUDENT` where the `Grade` is greater than 80. [3]

8.6 Explain *why* aiming for 3NF (each fact stored once) makes a database safer to update. [2]

8.7 Challenge. Write a SQL query that counts how many rows are in a `CUSTOMER` table. [2]

6 A company is developing a website that will allow users to create an account and then play a quiz every day. The data about the users and the quizzes are stored in a database.

A user must select a unique username and enter a valid email address to create an account. All users must be over the age of 16. A new quiz is given to the users every day. Each quiz is stored in its own text file.

The database stores the filename of each quiz and the date it can be played. The user gets a score for each quiz they complete, which is stored in the database. The scores are used to give each user a rating, for example Gold.

(a) Create a 3-table design for this database normalised to Third Normal Form (3NF).

Give your table design in the format:

`TableName(PrimaryKey, Field1, Field2, …)`

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) The company is using a Database Management System (DBMS) to set up the database.

Describe what is meant by the following DBMS features:

Data dictionary .....

.....

.....

.....

Logical schema .....

.....

.....

.....

[4]

(c) The company has another database, FARMING, for a different game.

The database FARMING has a table named EVENT which is shown with some sample data.

PlayerID	EventID	Category	Points
000123	3	Build	100
000124	1	Grow	36
000123	4	Grow	22
000123	7	Create	158
000125	3	Grow	85
000125	4	Build	69

(i) The database FARMING has a second table created named PLAYER that has the primary key PlayerID.

The field PlayerID in EVENT needs to be set up as a foreign key to link to PlayerID in PLAYER.

Write a Structured Query Language (SQL) script to change the table definition for EVENT to link the foreign key to PLAYER.

.....

.....

.....

..... [2]

(ii) Write an SQL script to return the number of events that each player has completed.

.....

.....

.....

.....

.....

..... [3]

4 An assessment board wants to store the marks students achieved in exams in a database named RECORDS.

Part of the database design includes these two tables:

EXAM(ExamID, Subject, Level, TotalMarks)

EXAM_QUESTION(ExamQuestionID, ExamID, QuestionNumber, Question, MaxMark)

(a) Identify the relationship between EXAM and EXAM_QUESTION.

.....

..... [1]

(b) Sample data for the table EXAM is shown:

ExamID	Subject	Level	TotalMarks
00956124	Computer Science	2	75
00956125	Computer Science	3	120
00956126	Mathematics	2	100
00956127	Mathematics	3	150
00956128	Physics	2	70
00956129	Physics	3	80

Write a Structured Query Language (SQL) script to define the table EXAM.

.....

.....

.....

.....

.....

..... [3]

(c) The table `EXAM_QUESTION` has been created but the foreign key has not been linked.

Write an SQL script to update `EXAM_QUESTION` **and** link the foreign key to `EXAM`.

.....

.....

.....

..... [2]

(d) The database also needs to store data about the students, the exams the students have taken and the marks the students achieved in each question of each exam.

Describe the additional tables that will need to be included in the database **and** explain how all the tables in the database will be linked.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

9 Algorithm Design and Problem-solving – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
Computational thinking	计算思维			
decomposition	分解			
abstraction	抽象			
algorithm	算法			
sequence	顺序			
selection	选择			
iteration	迭代			
flowchart	流程图			
pseudocode	伪代码			
variable	变量			

Recall

You have solved problems step by step before. Bring this with you:

- A clear set of steps can be followed by someone else to get the same result.
- You have read simple flowcharts.

At AS we learn to think like a computer scientist and to write algorithms carefully — the planning that comes before any code.

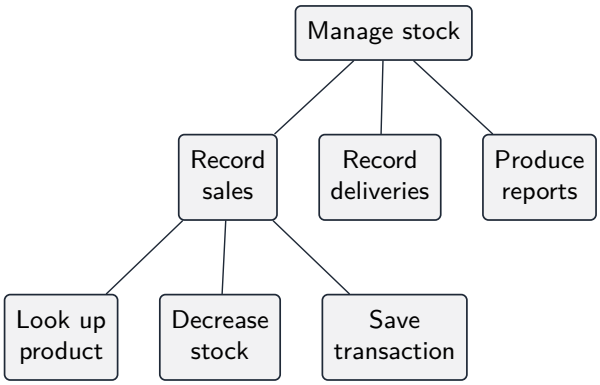
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Computational thinking

Computational thinking 计算思维 means solving a problem in a clear, step-by-step way that a computer could follow. Two key skills:

- **decomposition** 分解: break a big problem into smaller, easier parts;
- **abstraction** 抽象: keep only the details that matter and ignore the rest.

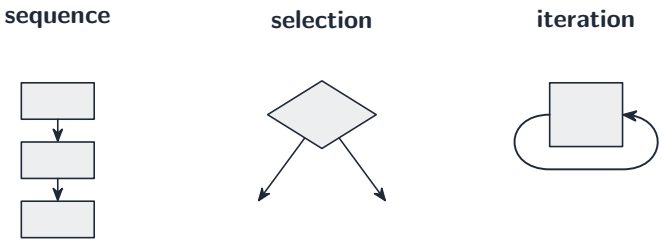


Worked example. To build a quiz app, decomposition splits it into "ask a question", "check the answer" and "keep the score"; abstraction ignores the screen colour and font, which do not affect how it works.

2 Algorithms and the three building blocks

An **algorithm** 算法 is a set of clear, ordered steps that always finishes. Every algorithm is built from just three structures:

- **sequence** 顺序: steps done in order;
- **selection** 选择: choose between paths with `IF`;
- **iteration** 迭代: repeat steps with a loop.

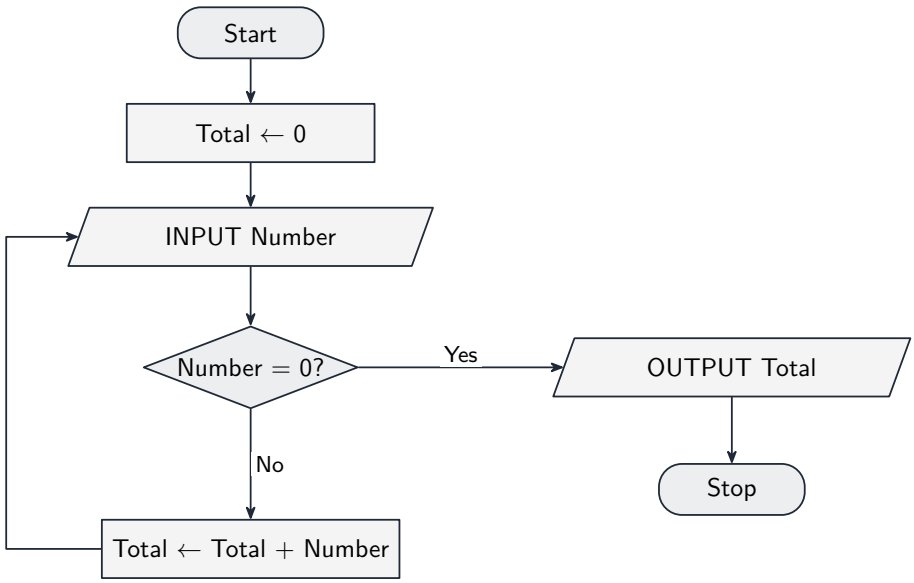


Worked example. "Keep asking for a password until it is correct" uses all three: read the input (sequence), check it (selection), and repeat while it is wrong (iteration).

Watch out: an algorithm must always *finish*. A loop needs a condition that eventually becomes false — or it runs forever.

3 Writing it down

We plan an algorithm as a **flowchart** 流程图 or in **pseudocode** 伪代码 (code-like English).



Pseudocode stores values in a **variable** 变量:

```
INPUT age
IF age >= 18 THEN
    OUTPUT "adult"
ELSE
    OUTPUT "minor"
ENDIF
```

Worked example. If `age` is 20, the test `age >= 18` is true, so this outputs "adult".

Practice

9.1 Explain what decomposition means. [1]

9.2 Explain what abstraction means. [1]

9.3 State which building block an `IF … ELSE` statement is. [1]

9.4 Name the three building blocks of any algorithm. [3]

9.5 Write pseudocode that inputs two numbers and outputs their sum. [3]

9.6 Explain why breaking a large program into smaller parts (decomposition) makes it easier to write and test. [2]

9.7 Challenge. Write pseudocode that keeps inputting numbers until a 0 is entered, then outputs their total. [3]

7 A coffee shop owner wants to introduce a computerised loyalty card system.

A programmer discusses the details of the system with the shop owner.

(a) Identify the stage of the program development life cycle that this discussion is part of **and** give a document that will be produced during this stage.

Stage .....

Document .....

.....

[2]

(b) The shop will give each customer a loyalty card that displays a unique customer ID as a bar code. A customer will be able to present their card each time they make a purchase. The system will scan the bar code, calculate points, and add them to the customer's total. When the customer next makes a purchase and presents their card, they will have the option to exchange points for a discount.

The designer decides that this activity will be handled by a new module. Decomposition will be used to break the problem of designing the new module down into sub-problems (sub-modules).

Identify **four** sub-modules that could be used in the design of the new module **and** describe their use.

Sub-module 1 .....

Use .....

.....

Sub-module 2 .....

Use .....

.....

Sub-module 3 .....

Use .....

.....

Sub-module 4 .....

Use .....

.....

[4]

9 Algorithm Design and Problem-solving – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
flowchart	流程图			
trace table	追踪表			
Stepwise refinement	逐步求精			
linear search	线性查找			

Recall

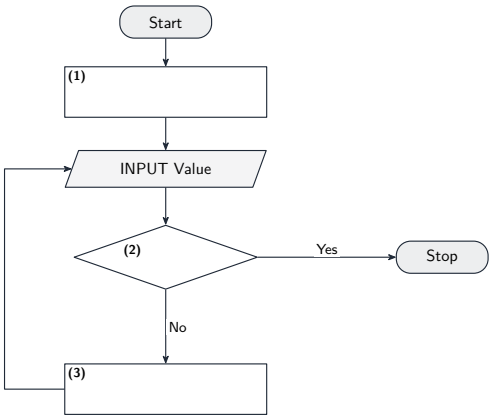
In **Part 1** you met computational thinking (decomposition, abstraction), the three constructs (sequence, selection, iteration), and pseudocode. Part 2 adds drawing and checking flowcharts, refining a design step by step, and a standard searching algorithm.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 Flowcharts and trace tables

A **flowchart** 流程图 draws an algorithm with standard shapes: a rounded box = start/stop, a parallelogram = input/output, a rectangle = a process, a diamond = a decision, and arrows = the flow.



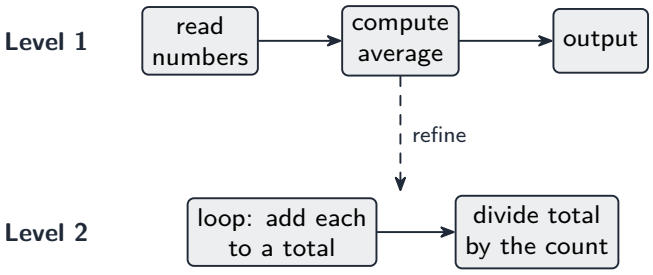
To check an algorithm, make a **trace table** 追踪表 — a column for each variable — and fill in the values row by row as you "run" the steps by hand.

Worked example. Tracing `total ← 0; FOR i ← 1 TO 3; total ← total + i`: the rows give `total` $= 0, 1, 3, 6$, so the final total is 6.

■ 2 Stepwise refinement

Stepwise refinement 逐步求精 starts with a high-level outline and expands each step until it can be coded:

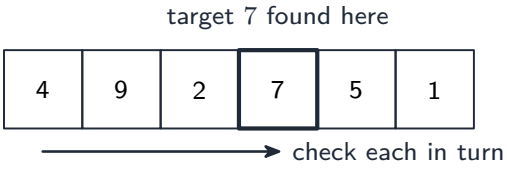
- *Level 1:* read the numbers → compute the average → output it;
- *Level 2:* the input loop that adds each number to a running total, then dividing by the count.



Worked example. "Make a sandwich" refines to "get bread → add filling → close it"; then "add filling" refines further into "spread butter → add cheese".

■ 3 A standard algorithm: linear search

A **linear search** 线性查找 checks each element in turn until it finds the target (or reaches the end). It is simple and works on any list, but is slow on a long list.




```
FOR i ← 1 TO n
    IF A[i] = Target THEN
        OUTPUT "Found at ", i
    ENDIF
NEXT i
```

Worked example. Searching `[4, 9, 2, 7]` for 7: check 4 (no), 9 (no), 2 (no), 7 (yes) — found at position 4.

Watch out: when you fill a trace table, follow the steps *exactly* as written — do not silently "fix" the algorithm in your head, or you will miss the very bug you are hunting.

Practice

9.1 State the flowchart shape used for (a) a decision, (b) an input or output. [2]

9.2 State one advantage of a linear search over a more complex search. [1]

9.3 Trace this code and give the final value of `x`: `x ← 0`; `FOR i ← 1 TO 4`; `x ← x + i`; `NEXT i`. [2]

9.4 Explain what stepwise refinement means. [2]

9.5 State how a linear search works, and when it is slow. [2]

9.6 Explain why a trace table is useful for finding a logic error that does *not* crash the program. [2]

9.7 Challenge. A linear search looks through a list of 1000 items. State the largest number of comparisons it might need, and explain when that happens. [2]

2 An algorithm will:

1. prompt and input a sequence of 100 integer values, one at a time
2. sum the positive integers
3. output the result of the sum.

(a) Write pseudocode for the algorithm.

Assume the value zero is neither positive nor negative.

You must declare all variables used in the algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(b) The algorithm requires the use of basic constructs. One of these is sequence.

Identify **one other** basic construct required by the algorithm **and** describe how it is used.

Construct .....

Use .....

.....

[2]

10 Data Types and Structures – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
data type	数据类型			
record	记录			
array	数组			
element	元素			
index	索引			
file	文件			
abstract data type	抽象数据类型			
stack	栈			
queue	队列			
linked list	链表			

Recall

At IGCSE you stored single values in variables. Bring this with you:

- A variable holds one value (a number, a character, …).
- You worked with lists of items informally.

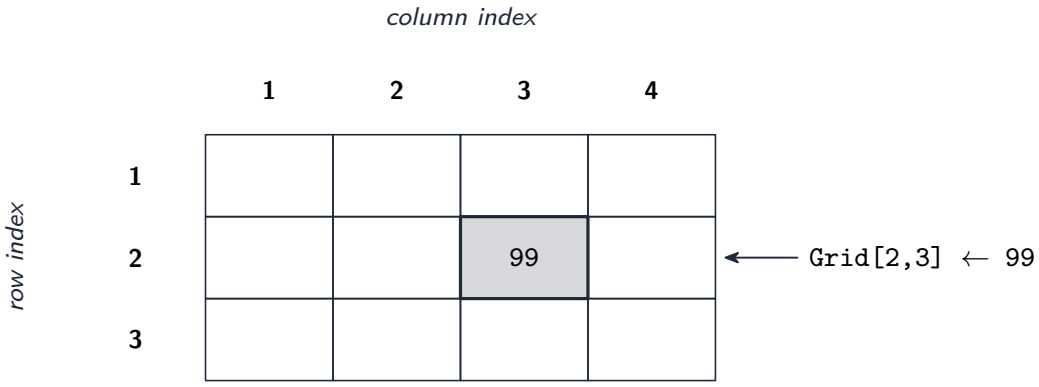
At AS we store whole collections of data and meet some useful data structures.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Data types, records and arrays

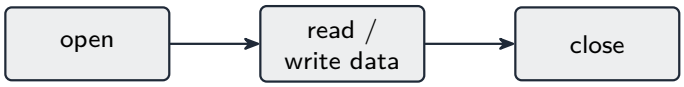
- a **data type** 数据类型 says what kind of value is stored (integer, real, char, string, Boolean).
- a **record** 记录 groups several fields of *different* types under one name (like one row of a table).
- an **array** 数组 stores many values of the *same* type in one list; you reach each **element** 元素 by its **index** 索引 (its position). A 2D array is a grid of rows and columns.



Worked example. `scores = [70, 85, 90]` is an array (all integers); a `Student` record holds `name` (string), `age` (integer) and `passed` (Boolean) together in one structure.

2 Files

A **file** 文件 stores data on secondary storage, so it is kept after the program ends. You open a file, read or write data, then close it.



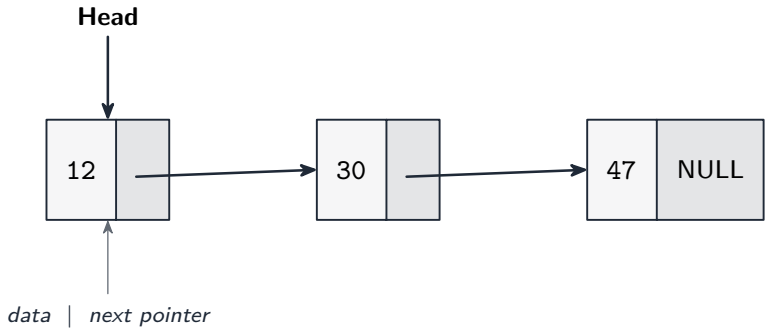
Worked example. A game saves its high-score table to a file when it closes, so the scores are still there the next time you play.

3 Abstract data types: stacks and queues

An **abstract data type** 抽象数据类型 (ADT) is described by what it *does*, not how it is stored:

- a **stack** 栈 is last-in, first-out (LIFO) — like a pile of plates;
- a **queue** 队列 is first-in, first-out (FIFO) — like a line of people.

A **linked list** 链表 stores items where each one points to the next.



Worked example. Pressing "undo" again and again uses a stack — your most recent action is undone first; a printer queue uses a queue — the first job sent is printed first.

Watch out: in many languages an array index starts at 0, not 1 — so `A[0]` is the first element. Always check where the counting begins.

Practice

10.1 Give two examples of a data type. [2]

10.2 An array `A` holds the values $10, 20, 30, 40$ in positions 1 to 4. State the value at index 2. [1]

10.3 A stack is LIFO and a queue is FIFO. State which one removes the **most recently added** item first. [1]

10.4 State one difference between an array and a record. [2]

10.5 State one thing a file lets you do that an ordinary variable does not. [1]

10.6 Explain *why* a stack is the right structure for an "undo" feature. [2]

10.7 Challenge. Items `A`, `B`, `C` are pushed onto a stack in that order, then two items are popped. State which two are removed, and in what order. [2]

3 A program is needed to manage individual rentals in a car-hire business.

The data items for each rental will be held in a record structure of type `RentalRecord`
The programmer has started to define the items that will be needed:

Item	Example value	Comment
`RentalID`	`"AB1234"`	a unique alpha-numeric value
`CarID`	`241`	a numeric value used as an array index
`DisCode`	`'S'`	a letter indicating the type of discount offered
`Start`	`13/06/2025`	when the rental starts
`Duration`	`7`	the number of days of the rental
`Completed`	`FALSE`	`TRUE` when the car is returned and the rental charge paid

(a) (i) Write pseudocode to declare the record structure for type `RentalRecord`

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) A 1D array `Rental` containing 500 elements is used to store the data for all rental records.

Write pseudocode to declare the `Rental` array.

.....

..... [2]

(b) State **three** benefits of using an array of records to store the data for all rentals.

1 .....

.....

2 .....

.....

3 .....

....

[3]

4 A global 1D array `Data` contains 100 elements of type integer.

A function `Check()` will:

- total the element values in odd index locations (1, 3, 5 ... 97, 99)
- total the element values in even index locations (2, 4, 6 ... 98, 100)
- return one of three strings 'Odd', 'Even' or 'Same' to indicate which total is the greater, or whether the totals are the same.

Write pseudocode for the function `Check()`.

[6]

10 Data Types and Structures – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
push	入栈			
pop	出栈			
overflow	溢出			
underflow	下溢			
enqueue	入队			
dequeue	出队			
circular array	循环数组			
end of file	文件结束			

Recall

In **Part 1** you met data types, records, arrays (1-D and 2-D), files, and three abstract data types — the stack, queue and linked list. Part 2 looks at the operations on a stack and a queue, and at reading and writing files.

New ideas

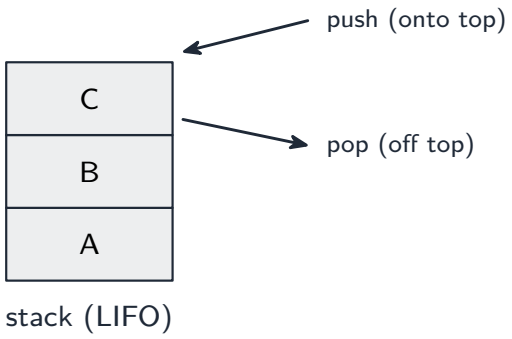
Read these first. Each idea has a short worked example. Then do the Practice.

1 Stack operations (LIFO)

A stack is **last-in, first-out**. Two operations:

- **push** 入栈 — add an item to the top;
- **pop** 出栈 — remove the item from the top.

Trying to push onto a full stack is **overflow** 溢出; trying to pop an empty stack is **underflow** 下溢. (Uses: undo history, function return addresses.)



Worked example. Start empty; `push(5)`, `push(8)` → the top is 8. `pop()` removes and returns 8, leaving 5 on the stack.

2 Queue operations (FIFO)

A queue is **first-in, first-out**. Two operations:

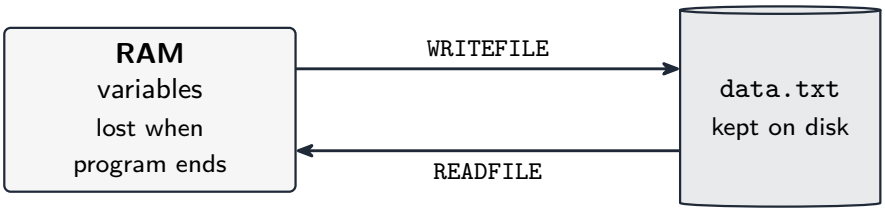
- **enqueue** 入队 — add an item at the rear;
- **dequeue** 出队 — remove the item from the front.

A **circular array** 循环数组 lets the front and rear pointers wrap back to the start of the array, so no space is wasted. (Uses: print queues, scheduling.)

Worked example. `enqueue(5)`, `enqueue(8)` → the front is 5. `dequeue()` removes and returns 5, leaving 8.

3 Handling files

Data in variables is lost when the program ends, so to keep it you write to a **file**. Open it, read or write, then close it; test for the **end of file** 文件结束 (EOF) while reading.




```
OPENFILE "data.txt" FOR READ
WHILE NOT EOF("data.txt") DO
    READFILE "data.txt", Line
    OUTPUT Line
ENDWHILE
CLOSEFILE "data.txt"
```

Worked example. The loop above prints every line of `data.txt`; the `NOT EOF` test stops it cleanly at the end instead of trying to read past the last line.

Watch out: check for EOF *before* each read — reading past the end of a file causes an error.

Practice

10.1 Items A, B, C are pushed onto a stack in that order, then one item is popped. State which item is removed. [1]

10.2 Items A, B, C are enqueued in that order, then one is dequeued. State which item is removed. [1]

10.3 State what is meant by stack (a) overflow and (b) underflow. [2]

10.4 State why a queue is usually stored in a circular array. [2]

10.5 State why a program must close a file after writing to it. [1]

10.6 Explain *why* a stack (not a queue) is used to remember where to return after each function call. [2]

10.7 Challenge. Items $1, 2, 3, 4$ are pushed onto a stack in that order. Three items are then popped and immediately enqueued onto a queue, in the order they were popped. One item is then dequeued. State which item comes out. [2]

3 An algorithm will output the **last** three lines from a text file `Result.txt`

The lines need to be output in the same order as they appear in the file.

Assume:

- Three variables `LineX`, `LineY` and `LineZ` will store the three lines. These are of type string and all three variables have been initialised to an empty string.
- The file exists and contains **at least** three lines.

(a) The algorithm to output the lines is expressed in eight steps.

Complete the steps.

1. Open the file .....
2. Loop until .....
3. ..... and store in `ThisLine`
4. Assign `LineY` to `LineX`
5. Assign `LineZ` to `LineY`
6. Assign `ThisLine` to `LineZ`
7. After the loop, ...
8. Output `LineX`, `LineY`, `LineZ`

[4]

(b) Explain the purpose of steps 4, 5 and 6 in the algorithm from part **(a)**.

.....

.....

..... [1]

(c) The requirement changes, and the algorithm will now output three lines from the file, starting from a **given** line number.

The modified algorithm will be implemented as a function which will:

- be called with an integer parameter representing the given line number
- output three lines, starting at the given line
- return `TRUE` if the 3 lines are output, or `FALSE` if it was not possible to output the 3 lines.

Describe the changes that need to be made to **steps 2 to 8** of the algorithm given in part **(a)**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

3 A program uses a stack to hold up to 30 integer values. The stack is implemented using a global integer variable and a global 1D array.

The array is declared in pseudocode:

```
DECLARE ThisStack : ARRAY[1:30] OF INTEGER
```

Stack design notes:

- The global variable `SP` acts as a stack pointer. `SP` contains the array index of the last value pushed onto the stack.
- If the stack is empty, then `SP` is assigned the value zero.
- The first item added to the stack will be stored in `ThisStack[1]`
- `SP` is incremented each time an item is added to the stack.

A function `Pop()` is written to remove an item from `ThisStack`. The function returns an item of type `PopData` which is defined in pseudocode:

```
TYPE PopData
   DECLARE Data   : INTEGER
   DECLARE Exists : BOOLEAN
ENDTYPE
```

The value removed from the stack is assigned to `Data` and `Exists` is set to `TRUE`. If it is **not** possible to remove a value from the stack, then `Exists` is set to `FALSE`

Complete the pseudocode for `Pop()`

```
FUNCTION Pop() RETURNS PopData

   DECLARE ThisPop : .....

   IF ..... THEN

      PopData.Exists ← ..... // Stack is empty

   ELSE

      PopData.Data ← .....

      PopData.Exists ← .....

      SP ← .....

   ENDIF

   RETURN ThisPop

ENDFUNCTION
```

[5]

11 Programming – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
variable	变量			
constant	常量			
assignment	赋值			
selection	选择			
iteration	迭代			
array	数组			
subroutines	子程序			
procedure	过程			
function	函数			
parameters	参数			
local variable	局部变量			
scope	作用域			

Recall

At IGCSE you wrote simple programs with input, output and `IF`. Bring this with you:

- A program follows your instructions in order.
- You used `IF` to make choices and loops to repeat.

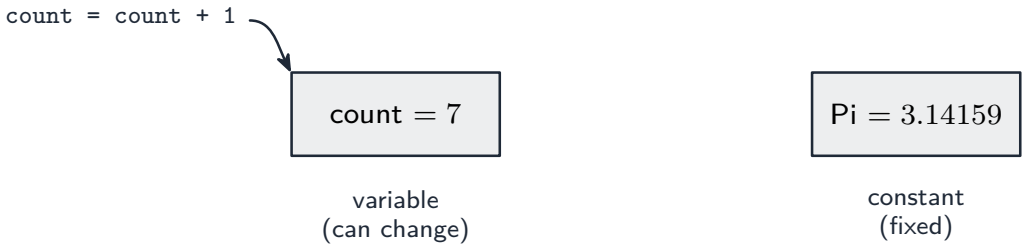
At AS we use variables carefully and break programs into reusable subroutines.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Variables, constants and assignment

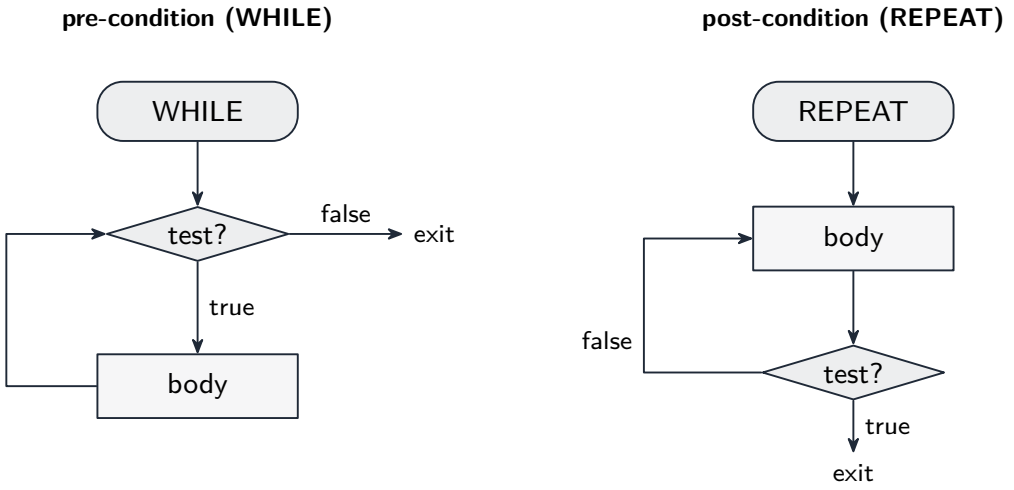
- a **variable** 变量 is a named store whose value can change; a **constant** 常量 keeps the same value all through the program.
- **assignment** 赋值 puts a value into a variable, written with an arrow: `count ← count + 1`.



Worked example. `VAT ← 0.20` suits a constant (it rarely changes); `count ← count + 1` is assignment — it reads the old `count`, adds one, and stores the result back.

2 The three constructs

- **selection** 选择 (`IF` / `CASE`) chooses between paths.
- **iteration** 迭代 repeats: a count-controlled loop (`FOR`) repeats a fixed number of times; a condition-controlled loop (`WHILE`) repeats until a condition changes.



An **array** 数组 is often processed with a `FOR` loop, one element at a time.

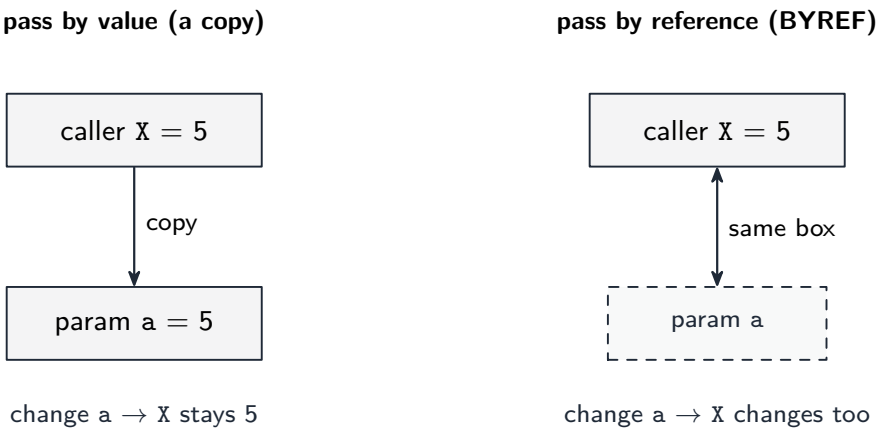
Worked example. To total 10 marks, use `FOR i ← 1 TO 10` (you know it is 10); to keep asking for a password, use `WHILE wrong` (you do not know how many tries it takes).

■ 3 Subroutines

Break a large program into named **subroutines** 子程序:

- a **procedure** 过程 carries out a job;
- a **function** 函数 carries out a job **and returns a value**.

Values passed into a subroutine are **parameters** 参数.



A **local variable** 局部变量 exists only inside its own subroutine — that area is its **scope** 作用域.

Worked example. `area(w, h)` is a function: it takes parameters `w` and `h` and *returns* `w * h`; `printReport()` is a procedure: it does a job but returns nothing.

> **Watch out:** a function *returns* a value you can use (`x ← area(3, 4)`); a procedure returns nothing, so `x ← printReport()` makes no sense.

Practice

11.1 State one difference between a variable and a constant. [2]

11.2 State what the assignment `total ← total + 5` does. [1]

11.3 State when you would use a `FOR` loop rather than a `WHILE` loop. [2]

11.4 State one difference between a procedure and a function. [2]

11.5 Explain what is meant by the scope of a local variable. [1]

11.6 Explain one advantage of breaking a large program into subroutines. [2]

11.7 Challenge. Write pseudocode for a function `bigger(a, b)` that returns the larger of the two numbers. [3]

1 **(a)** The table contains pseudocode examples.

Each example may contain statements that relate to one or more of:

- selection
- iteration (repetition)
- subroutines (procedures or functions).

Complete the table by placing **one or more** ticks ('✓') in each row.

Pseudocode example	Selection	Iteration	Subroutine

`IF Status = FALSE THEN`
`  FOR Count ← 1 TO 20`
`  CALL Reset(Count)`
`  NEXT Count`
`ENDIF`			

`OTHERWISE : Status ← TRUE`			
`WHILE AllDone() = TRUE`			
`30 : NextChar ← 'X'`			

[4]

(b) Complete the table by giving the appropriate data type.

Variable	Example data value	Data type
`Result`	`5.42`	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	`15/11/2009`	

[3]

(c) Evaluate each expression in the table by using the data values shown in **(b)**.

Write 'ERROR' if the expression contains an error.

Expression	Evaluates to
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) – 2, 1)`	

[4]

11 Programming – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)
operators	运算符	
library routines	库例程	
concatenation	拼接	
nested	嵌套	

Recall

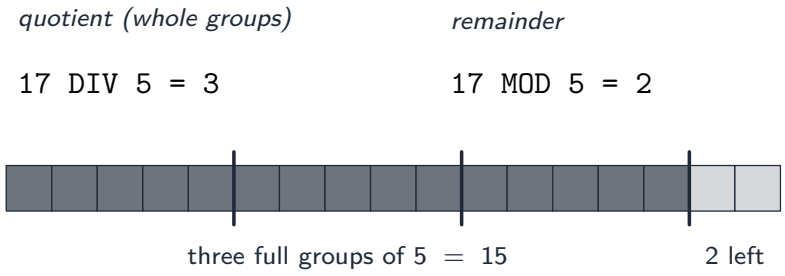
In **Part 1** you used variables and constants, the three constructs (selection, iteration), arrays, and subroutines (procedures, functions, parameters, scope). Part 2 adds two more operators and ready-made functions, the `CASE` statement, and writing clean, efficient code.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 Operators and library routines

Besides + − × ÷, two useful **operators** 运算符 work on whole numbers: `DIV` (integer division — `7 DIV 2 = 3`) and `MOD` (the remainder — `7 MOD 2 = 1`).

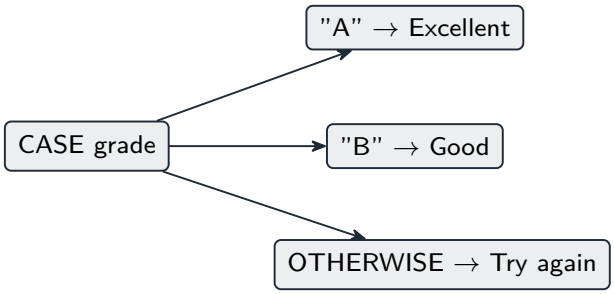


Many jobs already have ready-made **library routines** 库例程, so you don't write them yourself — string functions like `LENGTH(s)`, `LEFT(s, n)`, `MID(s, start, len)`, and joining two strings (**concatenation** 拼接) with `&`.

Worked example. `23 DIV 10 = 2` and `23 MOD 10 = 3`, so 23 is "2 tens and 3 ones"; `LEFT("HELLO", 2)` gives `"HE"`.

■ 2 The CASE statement

To test one value against many options, a deep **nested** 嵌套 `IF` is hard to read; a `CASE` is cleaner:




```
CASE OF grade
    "A": OUTPUT "Excellent"
    "B": OUTPUT "Good"
    OTHERWISE: OUTPUT "Try again"
ENDCASE
```

Worked example. With `grade = "B"`, the `CASE` above matches the `"B"` branch and outputs "Good".

■ 3 Writing efficient, readable code

- compute a value **once before a loop** if it doesn't change inside it;
- **exit a loop early** when the answer is found;
- use **meaningful names** (`numberOfPupils`, not `n`) and comment the *intent*, not the mechanics.

slower

```
FOR i = 1 TO n
  x = slow()  // every pass
  use(x, i)
NEXT i
```

faster

```
x = slow()  // once
FOR i = 1 TO n
  use(x, i)
NEXT i
```

Worked example. If a loop runs `tax ← price * 0.2` every pass but `price` never changes inside it, move that line above the loop so it runs once, not a thousand times.

Watch out: `DIV` throws away the remainder — `7 DIV 2` is 3, not 3.5. Use `MOD` when you want the remainder.

Practice

11.1 Find (a) `17 DIV 5` and (b) `17 MOD 5`. [2]

11.2 Find (a) `LENGTH("HELLO")` and (b) `LEFT("HELLO", 2)`. [2]

11.3 State when a `CASE` statement is a better choice than a nested `IF`. [1]

11.4 State the condition (using `MOD`) that is true when `n` is even. [1]

11.5 State one way to make a loop run more efficiently. [1]

11.6 Explain *why* `n MOD 2 = 0` is a correct test for whether `n` is even. [2]

11.7 Challenge. Using `DIV` and `MOD`, write pseudocode that inputs a number of seconds

and outputs the whole minutes and the leftover seconds. [3]

2 **(a)** A program contains a global 1D array of type `STRING` containing 65 elements.

An existing text file is used to store the data in the array. Only non-blank elements (those that do **not** contain an empty string) are written to the text file.

An algorithm will:
- write the first element of the array as a new line in the text file
- continue until all elements have been written, each to a new line of the text file.

Stepwise refinement is applied to the algorithm.

Describe up to **six** steps for this algorithm that could be used to produce pseudocode.

Do **not** use pseudocode statements in your answer.

Step 1 .....

.....

Step 2 .....

.....

Step 3 .....

.....

Step 4 .....

.....

Step 5 .....

.....

Step 6 .....

.....

[6]

(b) Iteration is one programming construct.

Identify **one** other programming construct that will be required when the algorithm from part **(a)** is converted into pseudocode and explain how it is used.

Construct .....

.....

Use .....

.....

[2]

6 In some countries, on the third Sunday in March, daylight saving time begins when clocks move forward by one hour.

A module `AdjustClock()` will take an integer parameter representing a year. The module will return an integer value representing the number of the day in March on which the clocks move forward.

For example, the following line of pseudocode would assign `DayNumber` the value 20:

```
DayNumber ← AdjustClock(2022)
```

Write pseudocode for the function `AdjustClock()`.

Date functions from the **insert** should be used in your solution.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

12 Software Development – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
development life cycle	开发生命周期			
requirements	需求			
waterfall	瀑布模型			
agile	敏捷			
prototype	原型			
syntax error	语法错误			
logic error	逻辑错误			
normal data	正常数据			
boundary data	边界数据			
maintenance	维护			

Recall

At IGCSE you wrote programs and fixed the mistakes in them. Bring this with you:

- A program rarely works perfectly the first time.
- Testing means trying the program to see if it behaves correctly.

At AS we look at how larger software is planned, tested and looked after.

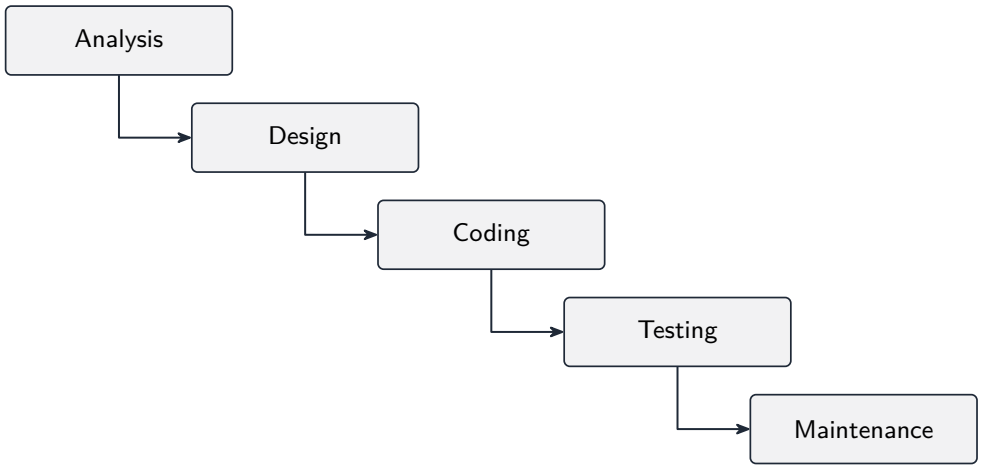
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 The development life cycle

Software is built in stages called the **development life cycle** 开发生命周期: find the **requirements** 需求, design, write the code, test, then maintain it. Two common models:

- **waterfall** 瀑布模型: each stage is finished before the next begins, in order;
- **agile** 敏捷: build in small repeated cycles, often making a quick **prototype** 原型 to show the user early and get feedback.



Worked example. A team builds a booking app the agile way: a basic prototype in two weeks, shown to the client, then improved in short cycles — instead of waiting months for one big waterfall release.

2 Testing

Programs can contain errors:

- a **syntax error** 语法错误 breaks the language's rules, so the program will not run;
- a **logic error** 逻辑错误 lets the program run but gives the wrong answer.

Test with different kinds of data: **normal data** 正常数据 (typical values that should be accepted), abnormal data (wrong values that should be rejected), and **boundary data** 边界数据 (values right at the edge of what is allowed).

Worked example. A field accepts ages 0–120: normal data 25 (accept), boundary data

0 and 120 (the edges), abnormal data −3 or 200 (reject).

■ 3 Maintenance

After release, software still needs **maintenance** 维护 — fixing bugs that appear, adapting it to new hardware or rules, and adding improvements.

Worked example. After release: fixing a crash is corrective; making it run on a new phone is adaptive; adding a dark-mode option is perfective.

Watch out: a logic error is the dangerous one — the program *runs*, so it looks fine, yet quietly gives wrong answers. A syntax error at least stops the program.

Practice

12.1 Name three stages of the development life cycle. [3]

12.2 State one difference between the waterfall and agile models. [2]

12.3 State the difference between a syntax error and a logic error. [2]

12.4 A program accepts ages from 0 to 120. Give one piece of boundary data and one piece of abnormal data. [2]

12.5 State what is meant by software maintenance. [1]

12.6 Explain *why* boundary data is especially good at finding errors in a program. [2]

12.7 Challenge. A program should accept marks from 1 to 100. Give one normal, one boundary and one abnormal test value, and say what each one checks. [3]

5 A program is developed to satisfy a specific customer requirement.

The project follows a program development life cycle model. This model divides the development process into several different stages.

(a) The table lists some of the development activities.

Complete the table by writing the name of the life cycle stage for each activity:

Activity	Name of life cycle stage
a structure chart is produced	
a program is modified to allow it to run on new hardware	
the programmer identifies the customer's requirements	
an Integrated Development Environment (IDE) provides context-sensitive help such as 'auto-complete'	

[4]

(b) Alpha and beta testing has been completed. The final testing stage is carried out by the customer.

Identify this final testing stage.

..... [1]

12 Software Development – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
structure chart	结构图			
state-transition diagram	状态转换图			
dry run	手工跟踪			
white-box testing	白盒测试			
black-box testing	黑盒测试			
integration testing	集成测试			
test plan	测试计划			
extreme data	极端数据			

Recall

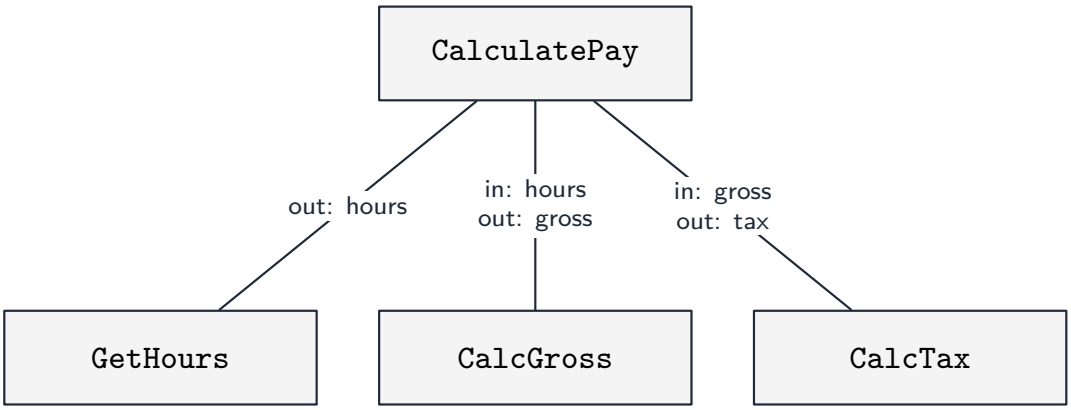
In **Part 1** you met the development life cycle and its models (waterfall, iterative, agile), the three error types (syntax, run-time, logic), test data (normal, abnormal, boundary) and the kinds of maintenance. Part 2 adds tools for *designing* a program, and methods for *testing* it.

New ideas

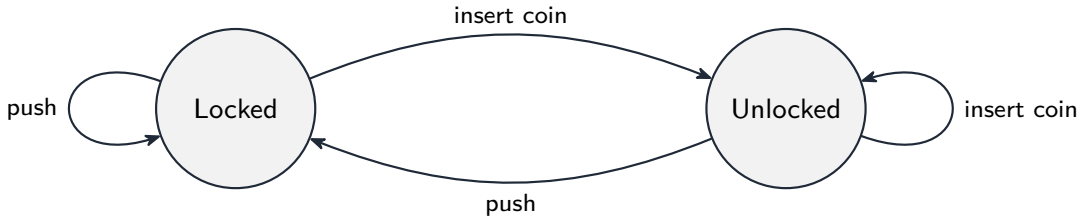
Read these first. Each idea has a short worked example. Then do the Practice.

■ 1 Design tools

A **structure chart** 结构图 shows a program broken into modules (subroutines), with the parameters passed between them — the caller above, the modules it calls below, and small arrows for the data going down and results coming back up.



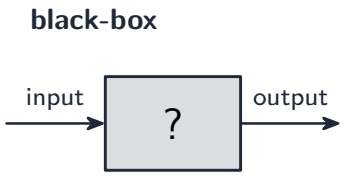
A **state-transition diagram** 状态转换图 shows the states a system can be in and the events that move it between them — useful for a vending machine, a lock, or a traffic light.



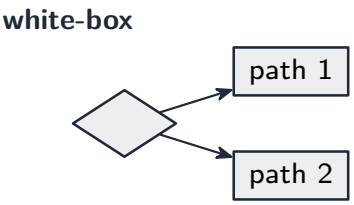
Worked example. A "make a drink" program's structure chart shows a top module calling `takeMoney`, `chooseDrink` and `dispense`, with the price passed down and an "ok?" result passed back up.

■ 2 Testing methods

- a **dry run** 手工跟踪 — trace the code by hand, writing each variable in a table;
- **white-box testing** 白盒测试 — designed from the *code's* structure, to cover every branch;
- **black-box testing** 黑盒测试 — designed from the *specification* only: give inputs, check the outputs;
- **integration testing** 集成测试 — combine the modules and test that they work together.



test from the spec



test every path in the code

Worked example. Black-box: a tester only knows "input two numbers, get their sum", so they try $2 + 3$ and expect 5. White-box: a tester who can see the code chooses inputs that make every `IF` branch run.

■ 3 Test strategy and test data

A test strategy is the overall plan; a **test plan** 测试计划 lists each test with its input, expected output and a column for the actual output. As well as normal, abnormal and boundary values, include **extreme data** 极端数据 — the largest and smallest values that are still accepted.

Watch out: black-box testing cannot tell you a hidden branch of code was never run — that is exactly what white-box testing checks.

Practice

12.1 State what a structure chart shows about a program. [2]

12.2 State what a state-transition diagram shows, and give one system it suits. [2]

12.3 State one difference between white-box and black-box testing. [2]

12.4 A field accepts marks from 0 to 100. Give one example each of (a) normal, (b) abnormal, (c) boundary data. [3]

12.5 State what integration testing checks. [1]

12.6 Explain *why* a project uses both black-box and white-box testing, rather than just one. [2]

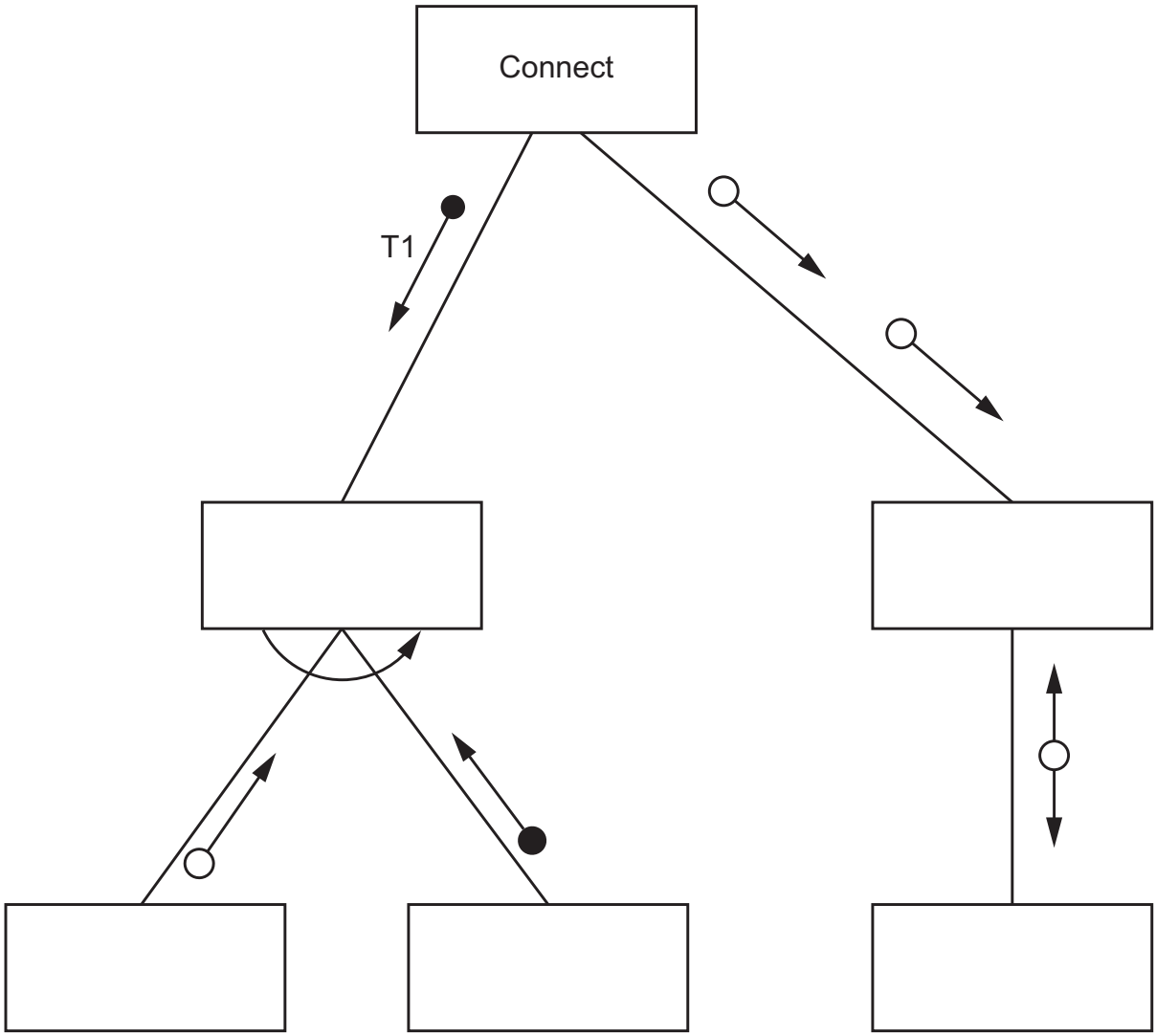
12.7 Challenge. A turnstile is either `Locked` or `Unlocked`. Describe, in words, a state-transition for it: name two events and state what each one changes. [2]

7 A program contains six modules with headers as follows:

Pseudocode module header
`PROCEDURE Connect()`
`FUNCTION Activate(H1 : STRING, Code : INTEGER) RETURNS BOOLEAN`
`FUNCTION Sync(T1 : BOOLEAN, S2 : REAL) RETURNS INTEGER`
`PROCEDURE Initialise(BYREF ID : INTEGER, BYVAL CC : INTEGER)`
`FUNCTION Reset(RA : STRING) RETURNS INTEGER`
`FUNCTION Enable(SA : INTEGER) RETURNS BOOLEAN`

Module `Connect()` will call either `Activate()` or `Sync()`. This is decided at run-time.

(a) Complete the structure chart for these modules.



[5]

(b) Explain the meaning of the curved arrow symbol used in the diagram in part **(a)**.

.....

..... [2]

5 A software developer follows a program development life cycle. The life cycle divides the development process into various stages.

(a) The following table lists some development activities.

Complete the table by writing the name of the life cycle stage for each activity.

Activity	Name of life cycle stage
A compiler is used.	
A program that has been released for general use is modified.	
The dry run method is used.	
The program structure is specified.	

[4]

(b) A software developer has written modules `Test_A()` and `Test_B()`. These have been written but contain errors. These modules are called from several places in the main program and testing of the main program (integration testing) has to stop.

Identify a method that can be used to continue testing the main program **before** the errors in these modules have been corrected **and** describe how this would work.

Method .....

Description .....

.....

.....

.....

[3]