Topic 07

IGCSE Computer Science

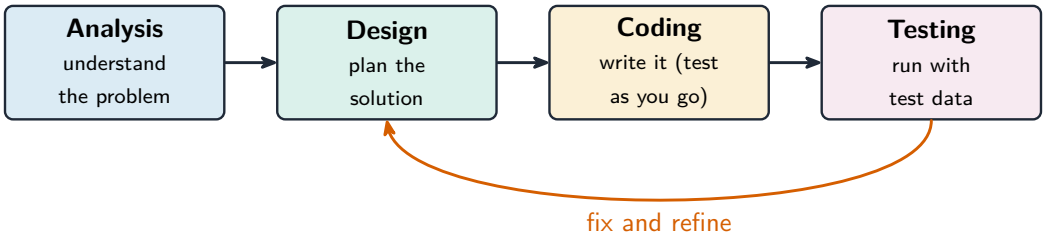
The program development life cycle

The **program development life cycle** 程序开发生命周期 is the set of stages used to make a program. There are four stages.

Software is written by programmers, who follow the development life cycle

Image: Crew crew, CC0 (commons.wikimedia.org)

Stage	What you do
analysis 分析	study the problem and work out what is needed
design 设计	plan how the program will work
coding 编码	write the program code and test it as you go
testing 测试	run the finished program with test data to find errors



The four stages of program development; testing feeds back to fix and refine the design

Tabel Simbol-simbol *flowchart* program

No.	Simbol	Keterangan
1		**Kotak Terminal (*Terminal Symbol/Terminator*)** ▫ simbol dimulainya (Start/Begin) dan diakhirinya suatu program (Finish/End).
2		**Kotak Proses (*Processing Symbol*)** ▫ tempat untuk menuliskan perintah/operasi aritmatika.
3		**Kotak Input dan Output (*Input/Output Symbol*)** ▫ tempat untuk memberikan masukan (*input*) atau menampilkan keluaran (*output*) dari suatu operasi.
4		**Kotak Keputusan (*Decision Symbol*)** ▫ tempat pertanyaan/pengujian kondisi, berisikan operasi perbandingan logika, dengan dua nilai keluaran (YA/BENAR/*TRUE* dan TIDAK/SALAH/*FALSE*).
5		**Lingkaran Penyambung (*Connector Symbol*)** ▫ digunakan jika diagram alir belum selesai dan akan berlanjut ke sampingnya pada halaman yang sama.
6		***Off-page Connector Symbol*** ▫ digunakan jika diagram alir belum selesai dan akan berlanjut ke halaman yang berbeda.
7		***Looping Symbol/Preparation Symbol*** ▫ digunakan untuk memberikan nilai awal pada suatu variabel/*counter* dan menggambarkan proses yang perulangan.
8		**Arah Proses (*Direction*)** ▫ untuk menunjukkan arah aliran proses program.

A program flowchart sets out the steps and decisions of a program during the design stage

Image: Mistervip.wa, CC BY-SA 4.0 (commons.wikimedia.org)

Analysis

In analysis you understand the problem. Two key skills help:

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

Design

In design you plan the solution, often using decomposition. You can show the parts as **sub-systems** 子系统 in a **structure diagram** 结构图 (a chart that splits a system into smaller boxes).

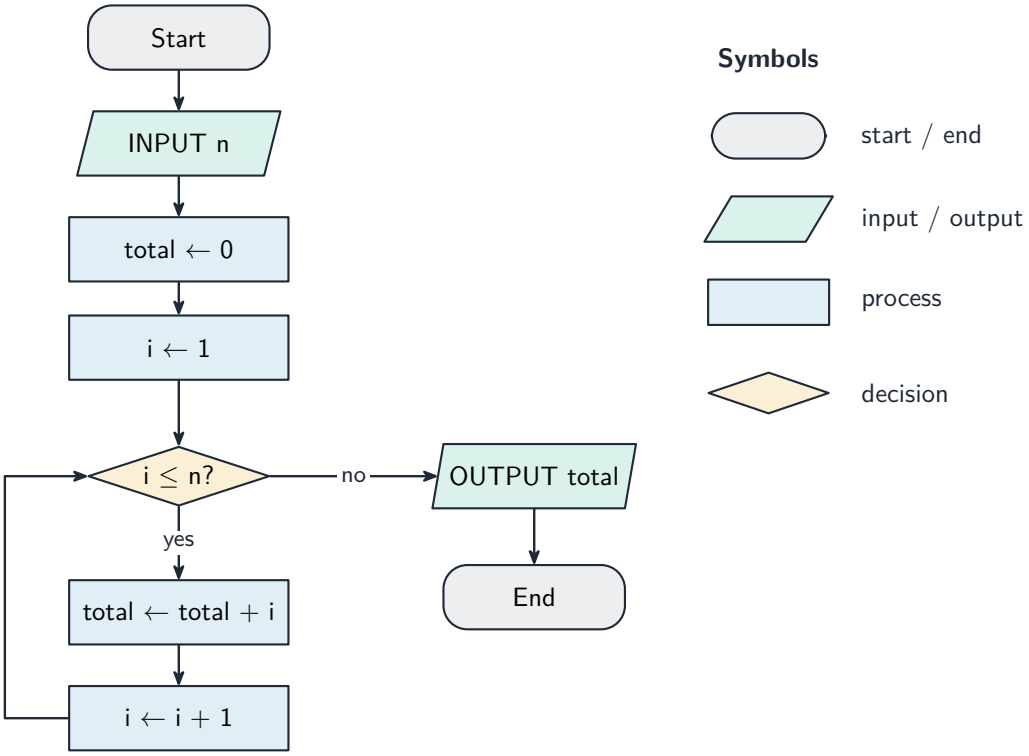
Coding and testing

In **coding** you write the program code. You use **iterative testing** 迭代测试—test small parts again and again as you build them. In **testing** you run the whole program with **test data** 测试数据 to check it works.

Design tools

You can plan a solution in three main ways.

- a **structure diagram** —shows the parts of a system and how they fit together;
- a **flowchart** 流程图—a diagram using boxes and arrows to show the steps in order;
- **pseudocode** 伪代码—steps written in simple, code-like English (not a real language).



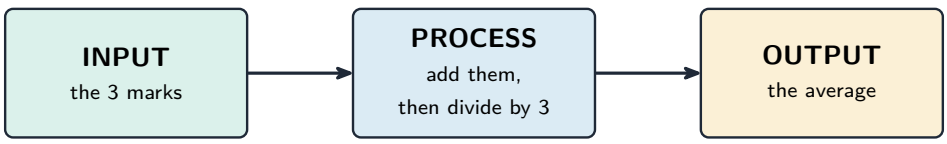
A flowchart for the sum algorithm, using the standard symbols (start/end, input/output, process, decision)

Algorithms

An **algorithm** 算法 is a set of steps, in the right order, that solves a problem. Every algorithm can be split into three parts:

- **input** 输入—the data that goes in;
- **processing** 处理—the work done on the data;
- **output** 输出—the result that comes out.

This is called decomposition into inputs, processes and outputs. For example, for ”find the average of three marks”: the inputs are the three marks; the processing is adding them and dividing by 3; the output is the average.



Every algorithm decomposes into input, processing and output —here, finding the average of three marks

Validation and verification

When data is entered, you check it to reduce mistakes.

Validation 验证 checks that the data is **sensible** and follows the rules. It cannot check that the data is true, only that it is allowed.

Validation check	What it checks
range check 范围检查	the value is between a lowest and highest allowed value
length check 长度检查	the number of characters is allowed (e.g. a password 8)
type check 类型检查	the data is the right type (e.g. a number, not letters)
presence check 存在性检查	something has actually been entered (not left blank)
format check 格式检查	the data is in the right pattern (e.g. a date as dd/mm/yyyy)
check digit 校验码	an extra digit confirms a number was entered correctly

Verification 核实 checks that data was **copied or entered correctly** (no mistakes while typing it in). Two methods:

- **visual check** 目视检查—a person compares the typed data with the original;
- **double entry** 双重输入—the data is entered twice and the two copies are compared.

Trace tables

A **trace table** 追踪表 records the value of each variable as an algorithm runs, step by step. It helps you:

- check that an algorithm works correctly;
- work out **what an algorithm does** by following it with given data.

Example: trace this algorithm with the input 5.

```
INPUT n
total ← 0
FOR i ← 1 TO n
    total ← total + i
NEXT i
OUTPUT total
```

i	total	OUTPUT
1	1	
2	3	
3	6	
4	10	
5	15	15

The trace shows the algorithm adds up 1 to n. With input 5 the output is 15.

Test data

Test data is data you use to test a program. There are four types you must know.

Type	Meaning	Example (age 0–120 allowed)
normal 正常数据	sensible data that should be accepted	25
abnormal 异常数据	wrong data that should be rejected	-4 or "cat"
extreme 极端数据	the largest and smallest values still allowed	0 and 120
boundary 边界数据	the values on each side of a limit (one allowed, one not)	120 and 121

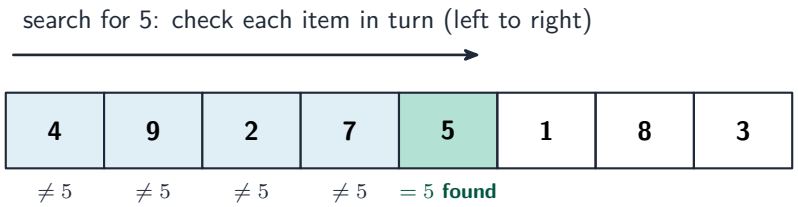
Standard methods of solution

You must know these common algorithms.

Linear search

A **linear search** 线性查找 checks each item in a list, one by one, until it finds the value it wants or reaches the end.

```
found ← FALSE
FOR i ← 0 TO 9
    IF list[i] = searchValue THEN
        found ← TRUE
    ENDIF
NEXT i
OUTPUT found
```

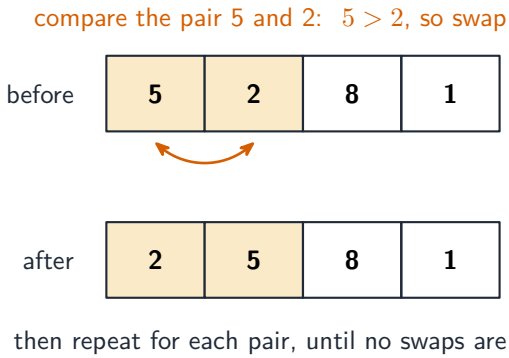


Linear search checks each item in turn from the start until it finds the value

Bubble sort

A **bubble sort** 冒泡排序 puts a list in order. It compares each pair of side-by-side items and swaps them if they are in the wrong order. It repeats this until no more swaps are needed.

```
FOR i ← 0 TO 8
    IF list[i] > list[i + 1] THEN
        temp ← list[i]
        list[i] ← list[i + 1]
        list[i + 1] ← temp
    ENDIF
NEXT i
```



Bubble sort compares each side-by-side pair and swaps them if they are out of order, repeating until sorted

Totalling and counting

- **totalling** 求和—keep adding values to a running total (`total ← total + value`).
- **counting** 计数—add 1 to a counter each time something happens (`count ← count + 1`).

Maximum, minimum and average

- to find the **maximum** 最大值: keep the largest value seen so far.
- to find the **minimum** 最小值: keep the smallest value seen so far.
- to find the **average** 平均值: divide the total by how many values there are.

```
total ← 0
FOR i ← 0 TO 9
    total ← total + list[i]
NEXT i
average ← total / 10
OUTPUT average
```