# 07 Algorithm Design and Problem-Solving

SrWL Learning · Complete Classroom Teaching Version · 2026–2028 target

## Plan and test a solution

55 editable slides / Explanation · Visual · Practice · Teacher Check

# By the end of this chapter

- Development life cycle.
- Decomposition and design.
- Pseudocode and flowcharts.
- Searching, sorting and testing.

Learning goals are organised into bite-sized lessons.

# Our learning route

10 mini-lessons · each includes one guided check

01 Program development life cycle: A

02 Program development life cycle: B

03 Decomposition and abstraction: A

04 Decomposition and abstraction: B

05 Sequence, selection, iteration: A

06 Sequence, selection, iteration: B

07 Standard algorithms and trace tables: A

08 Standard algorithms and trace tables: B

09 Validation, verification and testing: A

10 Validation, verification and testing: B

# 01 Program development life cycle · Foundations

One clear idea to explain before moving to practice

**Analysis determines what the solution must do.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Program development life cycle — key points

- Analysis determines what the solution must do.
- Design describes how the solution will work.

Worked context: Maintenance follows deployment in a full development cycle.

# Program development life cycle — connect the ideas

Follow the concept from left to right

**01**

**Analysis determines what the solution must do.**

→

**02**

**Design describes how the solution will work.**

Apply the principles to this example: Maintenance follows deployment in a full development cycle.

# Try it · Program development life cycle

Think (30 s) · Pair discussion (1 min) · Share

**Explain the meaning of this idea and give a relevant example:**

**Design describes how the solution will work.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Program development life cycle

Model response · Correct misconceptions

**Design describes how the solution will work.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Maintenance follows deployment in a full development cycle.

# 02 Program development life cycle · Applications

One clear idea to explain before moving to practice

**Coding implements the designed solution.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Program development life cycle — key points

- Coding implements the designed solution.
- Testing checks behaviour using suitable test data.

Worked context: Maintenance follows deployment in a full development cycle.

# Program development life cycle — connect the ideas

Follow the concept from left to right

**01**

**Coding implements the designed solution.**

→

**02**

**Testing checks behaviour using suitable test data.**

Apply the principles to this example: Maintenance follows deployment in a full development cycle.

# Try it · Program development life cycle

Think (30 s) · Pair discussion (1 min) · Share

**Describe when this principle applies:**

**Testing checks behaviour using suitable test data.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Program development life cycle

Model response · Correct misconceptions

**Testing checks behaviour using suitable test data.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Maintenance follows deployment in a full development cycle.

# 03 Decomposition and abstraction · Foundations

One clear idea to explain before moving to practice

**Decomposition divides a complex task into smaller parts.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Decomposition and abstraction — key points

- Decomposition divides a complex task into smaller parts.
- Abstraction retains essential details and removes distractions.

Worked context: Break down a school library loan system into sub-systems.

# Decomposition and abstraction — connect the ideas

Follow the concept from left to right

**01**

**Decomposition divides a complex task into smaller parts.**

→

**02**

**Abstraction retains essential details and removes distractions.**

Apply the principles to this example: Break down a school library loan system into sub-systems.

# Try it · Decomposition and abstraction

Think (30 s) · Pair discussion (1 min) · Share

**Identify the key technical terms and describe their roles:**

**Abstraction retains essential details and removes distractions.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Decomposition and abstraction

Model response · Correct misconceptions

**Abstraction retains essential details and removes distractions.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Break down a school library loan system into sub-systems.

# 04 Decomposition and abstraction · Applications

One clear idea to explain before moving to practice

**Inputs, processes, outputs and storage describe a system.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Decomposition and abstraction — key points

- Inputs, processes, outputs and storage describe a system.
- Structure diagrams show a hierarchy of sub-systems.

Worked context: Break down a school library loan system into sub-systems.

# Decomposition and abstraction — connect the ideas

Follow the concept from left to right

**01**

**Inputs, processes, outputs and storage describe a system.**

→

**02**

**Structure diagrams show a hierarchy of sub-systems.**

Apply the principles to this example: Break down a school library loan system into sub-systems.

# Try it · Decomposition and abstraction

Think (30 s) · Pair discussion (1 min) · Share

**Compare this concept with a related alternative:**
**Structure diagrams show a hierarchy of sub-systems.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Decomposition and abstraction

Model response · Correct misconceptions

**Structure diagrams show a hierarchy of sub-systems.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Break down a school library loan system into sub-systems.

# 05 Sequence, selection, iteration · Foundations

One clear idea to explain before moving to practice

**Sequence: statements run in order.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Sequence, selection, iteration — key points

- Sequence: statements run in order.
- Selection: IF / ELSE or CASE chooses a path.

Worked context: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# Sequence, selection, iteration — connect the ideas

Follow the concept from left to right

**01**

**Sequence: statements run in order.**

→

**02**

**Selection: IF / ELSE or CASE chooses a path.**

Apply the principles to this example: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# Try it · Sequence, selection, iteration

Think (30 s) · Pair discussion (1 min) · Share

**Apply this idea to a new computer science scenario:**
**Selection: IF / ELSE or CASE chooses a path.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Sequence, selection, iteration

Model response · Correct misconceptions

**Selection: IF / ELSE or CASE chooses a path.**

EXAM TIP / EXTENSION

Use this worked context to add detail: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# 06 Sequence, selection, iteration · Applications

One clear idea to explain before moving to practice

**Iteration: FOR, WHILE or REPEAT repeats instructions.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Sequence, selection, iteration — key points

- Iteration: FOR, WHILE or REPEAT repeats instructions.
- Indentation makes the control structure clear.

Worked context: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# Sequence, selection, iteration — connect the ideas

Follow the concept from left to right

**01**

**Iteration: FOR, WHILE or REPEAT repeats instructions.**

→

**02**

**Indentation makes the control structure clear.**

Apply the principles to this example: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# Try it · Sequence, selection, iteration

Think (30 s) · Pair discussion (1 min) · Share

**Explain the meaning of this idea and give a relevant example:**

**Indentation makes the control structure clear.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Sequence, selection, iteration

Model response · Correct misconceptions

**Indentation makes the control structure clear.**

EXAM TIP / EXTENSION

Use this worked context to add detail: IF Score >= 50 THEN OUTPUT "Pass" ELSE OUTPUT "Fail" ENDIF.

# 07 Standard algorithms and trace tables · Foundations

One clear idea to explain before moving to practice

**Linear search checks items until a match is found.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Standard algorithms and trace tables — key points

- Linear search checks items until a match is found.
- Bubble sort compares and swaps neighbouring values.

Worked context: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# Standard algorithms and trace tables — connect the ideas

Follow the concept from left to right

**01**

**Linear search checks items until a match is found.**

→

**02**

**Bubble sort compares and swaps neighbouring values.**

Apply the principles to this example: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# Try it · Standard algorithms and trace tables

Think (30 s) · Pair discussion (1 min) · Share

**Describe when this principle applies:**

**Bubble sort compares and swaps neighbouring values.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Standard algorithms and trace tables

Model response · Correct misconceptions

**Bubble sort compares and swaps neighbouring values.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# 08 Standard algorithms and trace tables · Applications

One clear idea to explain before moving to practice

**Totals, counts, averages, min and max are common patterns.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Standard algorithms and trace tables — key points

- Totals, counts, averages, min and max are common patterns.
- A trace table records changing variables step by step.

Worked context: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# Standard algorithms and trace tables — connect the ideas

Follow the concept from left to right

**01**

**Totals, counts, averages, min and max are common patterns.**

→

**02**

**A trace table records changing variables step by step.**

Apply the principles to this example: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# Try it · Standard algorithms and trace tables

Think (30 s) · Pair discussion (1 min) · Share

**Identify the key technical terms and describe their roles:**
**A trace table records changing variables step by step.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Standard algorithms and trace tables

Model response · Correct misconceptions

**A trace table records changing variables step by step.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Trace: Total starts at 0; add 3, then 5 → Total = 8.

# 09 Validation, verification and testing · Foundations

One clear idea to explain before moving to practice

**Validation checks whether input meets rules.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Validation, verification and testing — key points

- Validation checks whether input meets rules.
- Verification checks whether data has been copied correctly.

Worked context: Example: age 0–120, test 0, 120, −1 and 121.

# Validation, verification and testing — connect the ideas

Follow the concept from left to right

**01**

**Validation checks whether input meets rules.**

→

**02**

**Verification checks whether data has been copied correctly.**

Apply the principles to this example: Example: age 0–120, test 0, 120, −1 and 121.

# Try it · Validation, verification and testing

Think (30 s) · Pair discussion (1 min) · Share

**Compare this concept with a related alternative:**
**Verification checks whether data has been copied correctly.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Validation, verification and testing

Model response · Correct misconceptions

**Verification checks whether data has been copied correctly.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Example: age 0–120, test 0, 120, −1 and 121.

# 10 Validation, verification and testing · Applications

One clear idea to explain before moving to practice

**Normal, abnormal and boundary test data test different cases.**

TEACHER EXPLANATION

Discuss why this principle matters in real computer systems.

# Validation, verification and testing — key points

- Normal, abnormal and boundary test data test different cases.
- Dry runs help identify logical errors before coding.

Worked context: Example: age 0–120, test 0, 120, −1 and 121.

# Validation, verification and testing — connect the ideas

Follow the concept from left to right

**01**

**Normal, abnormal and boundary test data test different cases.**

→

**02**

**Dry runs help identify logical errors before coding.**

Apply the principles to this example: Example: age 0–120, test 0, 120, −1 and 121.

# Try it · Validation, verification and testing

Think (30 s) · Pair discussion (1 min) · Share

**Apply this idea to a new computer science scenario:**
**Dry runs help identify logical errors before coding.**

YOUR TASK

Use accurate technical vocabulary. Add a second point or a specific example where possible.

# Teacher check · Validation, verification and testing

Model response · Correct misconceptions

**Dry runs help identify logical errors before coding.**

EXAM TIP / EXTENSION

Use this worked context to add detail: Example: age 0–120, test 0, 120, −1 and 121.

# Five essential ideas

Recall these without looking at the teaching slides

01 Program development life cycle

02 Decomposition and abstraction

03 Sequence, selection, iteration

04 Standard algorithms and trace tables

05 Validation, verification and testing

# End-of-chapter challenge

Answer in your notebook, then compare with a partner

**What loop suits exactly 20 repetitions? What test values check an age 1–18 input?**

TEACHER MODEL ANSWER

**FOR loop; test 1 and 18 (boundaries), valid middle and values outside range.**