Grade 13 A/L ICT · Lesson 9

9.2 Top-Down Design, Stepwise Refinement and Structure Charts

Syllabus 9.2. Big problems are easier when split into smaller parts. **Modularisation** divides a program into modules; **top-down design** starts with the whole task and **stepwise refinement** breaks each step into smaller steps until each can be coded. A **structure chart** shows the modules and how they call each other.

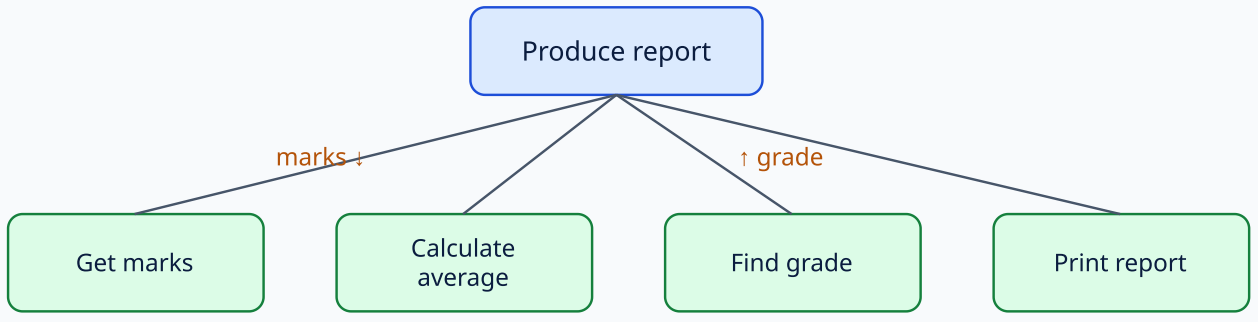
1. Stepwise refinement

Refining the task step by step

```
1 Produce a class report
  1.1 Get the marks
  1.2 Calculate the average
      1.2.1 Add all marks
      1.2.2 Divide by the number of marks
  1.3 Find the grade
      1.3.1 IF average >= 75 THEN A
      1.3.2 ELSE IF average >= 50 THEN C ELSE F
  1.4 Print the report
```

2. Structure chart

Structure chart (top-down design)



Each box is a module; lines show which module calls which; arrows on lines show data passed.

Structure chart for the class report

- Each **rectangle** is a module (a function in Python).
- **Lines** join a calling module to the modules it calls.
- Small arrows show **data** passed down (marks) and back up (grade).

3. Each module becomes a function

🐍 Top-down: a report built from small functions

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

```
def get_marks():
    return [78, 45, 88, 62, 39]

def average(marks):
    return sum(marks) / len(marks)

def grade(avg):
    if avg >= 75:
        return "A"
    elif avg >= 50:
        return "C"
    return "F"

def report():
    marks = get_marks()
    avg = average(marks)
    print("Average:", avg)
    print("Grade:", grade(avg))

report()
```

▶ Output

```
Average: 62.4
Grade: C
```

🧱 **Worked example: What does the modular program print, and which module decides it?** average() returns (78 + 45 + 88 + 62 + 39) ÷ 5 = **62.4**. grade() sees 62.4 ≥ 50 but < 75, so it returns **C**. report() prints both.

💡 Benefits of modules: easier testing, reuse, teamwork and simpler maintenance.

🚀 **Beyond the syllabus (2026 update)** *Not required for the exam – read for interest.* Large 2026 apps are split into microservices – the same top-down idea at the scale of whole servers.

Key Words

Modularisation – Dividing a program into separate modules.

Module – Self-contained part of a program that does one task.

Top-down design – Starting with the whole problem and dividing it into parts.

Stepwise refinement – Breaking each step into smaller, more detailed steps.

Structure chart – Diagram of modules and their calling hierarchy.

Main module – Top module that controls the sub-modules.

Sub-module – Module called by a higher module.

Data couple – Arrow on a structure chart showing data passed.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

Reuse – Using the same module in more than one place.

Function – Named block of code that can be called.

Bottom-up design – Building small parts first and combining them.

Decomposition – Breaking a complex problem into simpler ones.

Sample Q&A

1. What is stepwise refinement?

> **Answer:** Breaking each step into smaller steps until each can be coded.

2. What does a structure chart show?

> **Answer:** Modules, who calls whom and data passed.

3. Two benefits of modularisation?

> **Answer:** Easier testing and reuse.

4. Average in the modular program?

> **Answer:** 62.4.

5. Grade printed?

> **Answer:** C.

6. Top-down vs bottom-up?

> **Answer:** Top-down splits the whole; bottom-up builds parts then combines.

MCQ

1. Starting from the whole problem and splitting it is

A. top-down design
B. compiling
C. hand tracing
D. bottom-up design
E. debugging

2. Breaking steps into smaller detailed steps is

A. sorting
B. stepwise refinement
C. normalisation
D. interpretation
E. linking

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

3. A diagram of modules and calls is a

A. structure chart
B. trace table
C. flowchart symbol
D. Gantt chart
E. ER diagram

4. In Python a module of a structure chart is usually written as a

A. tuple
B. comment
C. function
D. constant
E. file mode

5. The modular program prints average

A. 60.4
B. 62.4
C. 65.0
D. 62.0
E. 78.0

6. The modular program prints grade

A. S
B. F
C. A
D. C
E. B

7. Arrows beside the lines of a structure chart show

A. data passed between modules
B. loop counters
C. file names
D. comments
E. syntax errors

8. Which is NOT a benefit of modularisation?

A. easier maintenance
B. reuse of code
C. teamwork
D. easier testing
E. it always makes programs run faster

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

9. Which term means: “Dividing a program into separate modules”?

A. Modularisation
B. Data couple
C. Main module
D. Top-down design
E. Bottom-up design

10. Which term means: “Self-contained part of a program that does one task”?

A. Decomposition
B. Module
C. Stepwise refinement
D. Reuse
E. Sub-module

11. Which term means: “Starting with the whole problem and dividing it into parts”?

A. Modularisation
B. Function
C. Structure chart
D. Top-down design
E. Data couple

12. Which term means: “Breaking each step into smaller, more detailed steps”?

A. Main module
B. Bottom-up design
C. Module
D. Reuse
E. Stepwise refinement

13. Which term means: “Diagram of modules and their calling hierarchy”?

A. Sub-module
B. Top-down design
C. Decomposition
D. Function
E. Structure chart

14. Which term means: “Top module that controls the sub-modules”?

A. Stepwise refinement
B. Main module
C. Data couple
D. Bottom-up design
E. Modularisation

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

15. Which term means: “Module called by a higher module”?

A. Decomposition
B. Structure chart
C. Reuse
D. Sub-module
E. Module

16. Which term means: “Arrow on a structure chart showing data passed”?

A. Modularisation
B. Top-down design
C. Data couple
D. Main module
E. Function

17. Which term means: “Using the same module in more than one place”?

A. Sub-module
B. Bottom-up design
C. Stepwise refinement
D. Reuse
E. Module

18. Which term means: “Named block of code that can be called”?

A. Decomposition
B. Top-down design
C. Function
D. Data couple
E. Structure chart

19. Which term means: “Building small parts first and combining them”?

A. Modularisation
B. Stepwise refinement
C. Bottom-up design
D. Reuse
E. Main module

20. Which term means: “Breaking a complex problem into simpler ones”?

A. Function
B. Structure chart
C. Sub-module
D. Module
E. Decomposition

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

Structured Questions

1. Explain top-down design and stepwise refinement. *(3 marks)*

> Whole to parts; refine each step; until codable.

2. Refine "calculate average" into smaller steps. *(3 marks)*

> Add all marks; count them; divide total by count.

3. Draw a structure chart for producing a class report. *(3 marks)*

> Main Produce report; Get marks, Calculate average, Find grade, Print report; data couples.

4. State three benefits of modularisation. *(3 marks)*

> Testing; reuse; teamwork/maintenance.

5. Write a Python function grade(avg) returning A, C or F. *(4 marks)*

> def grade(avg); if avg >= 75 return A; elif avg >= 50 return C; return F.

Summary

- Modularisation splits programs into modules.
- Top-down: whole problem first.
- Stepwise refinement adds detail step by step.
- Structure charts show modules and calls.
- Each module becomes a function.

MCQ answer key – 9.2

1. **A** – Top-down.
2. **B** – Refinement.
3. **A** – Structure chart.
4. **C** – Function.
5. **B** – 312 ÷ 5.
6. **D** – 50 ≤ 62.4 < 75.
7. **A** – Data couples.
8. **E** – Speed is not guaranteed.
9. **A** – Modularisation – Dividing a program into separate modules.
10. **B** – Module – Self-contained part of a program that does one task.
11. **D** – Top-down design – Starting with the whole problem and dividing it into parts.
12. **E** – Stepwise refinement – Breaking each step into smaller, more detailed steps.
13. **E** – Structure chart – Diagram of modules and their calling hierarchy.
14. **B** – Main module – Top module that controls the sub-modules.
15. **D** – Sub-module – Module called by a higher module.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

16. **C** – Data couple – Arrow on a structure chart showing data passed.
17. **D** – Reuse – Using the same module in more than one place.
18. **C** – Function – Named block of code that can be called.
19. **C** – Bottom-up design – Building small parts first and combining them.
20. **E** – Decomposition – Breaking a complex problem into simpler ones.

© 2026 schoolict.net (Ruwan Mangala Suraweera). All rights reserved.