Exercise walk-through

Programming concepts ▪ 8.1

eXercise

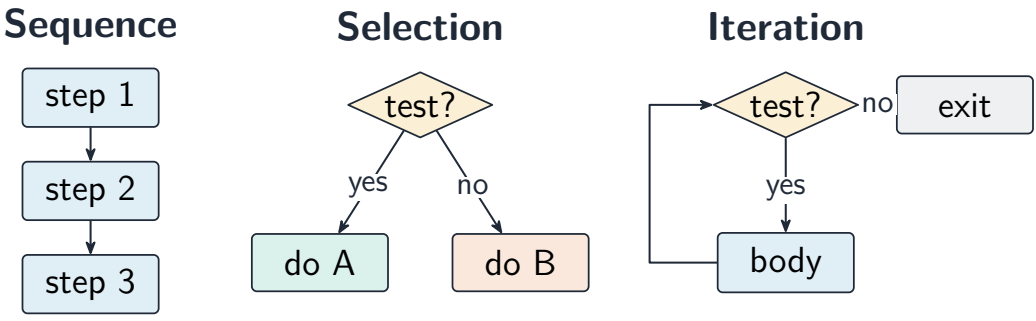
You must be able to

- declare variables/constants and use the five data types
- use sequence, selection (IF/CASE) and iteration (FOR/WHILE/REPEAT)
- use operators (MOD, DIV) and write procedures/functions

Method — Control structures

- `17 MOD 5` = 2 (remainder); `17 DIV 5` = 3 (whole part).

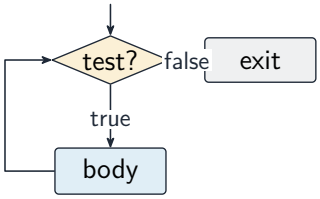
Method — Control structures



Sequence, selection, iteration flowcharts

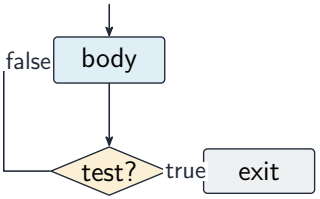
Method — Control structures

WHILE
(pre-condition)



tested first ⇒ may
run **zero** times

REPEAT
(post-condition)



tested last ⇒ runs
at least once

WHILE (test before) vs REPEAT (test after)

Method — Declaring, choosing and repeating

Declarations. A **variable** 变量 has a value that can change while the program runs; a **constant** 常量 is set once and never changes. In Cambridge pseudocode:

```
DECLARE Total : INTEGER
DECLARE Name  : STRING
CONSTANT VAT = 0.20
```

Method — Declaring, choosing and repeating

The five data types are `INTEGER` (whole numbers), `REAL` (numbers with a decimal part), `CHAR` (one character), `STRING` (text) and `BOOLEAN` (`TRUE` or `FALSE`).
Selection. `IF … THEN … ELSE … ENDIF` chooses between two paths; `CASE OF … OTHERWISE … ENDCASE` is neater when one variable is tested against many values.
Iteration, and how to choose:

Loop	When to use it	Runs at least once?
`FOR … TO … NEXT`	the number of repeats is known (count-controlled)	no, if the range is empty
`WHILE … DO … ENDWHILE`	the condition is tested **before** the body	no
`REPEAT … UNTIL`	the condition is tested **after** the body	yes

Method — Declaring, choosing and repeating

Worked example (count-controlled total).

```
DECLARE Total : INTEGER
DECLARE Mark  : INTEGER
DECLARE Count : INTEGER
Total ← 0
FOR Count ← 1 TO 10
    OUTPUT "Enter mark"
    INPUT Mark
    Total ← Total + Mark
NEXT Count
OUTPUT "The total is ", Total
```

Method — Declaring, choosing and repeating

Every mark in a "write pseudocode" question is earned by one of these: declaring and initialising, a correctly formed loop with the right count, the input inside the loop, the running total updated correctly, and a sensible output after the loop.

Operators. `MOD` gives the remainder and `DIV` the whole number of divisions: `17 MOD 5` is 2 and `17 DIV 5` is 3. Together they split a value, for example turning 137 seconds into $137 \text{ DIV } 60 = 2$ minutes and $137 \text{ MOD } 60 = 17$ seconds.

Procedures and functions. Both are named blocks of code that can be called from anywhere, which avoids repeating code. A **function** 函数 returns a value to the line that called it, so it is used inside an expression; a **procedure** 过程 does not return a value.

Practice 2.1 ▪ 2 marks

State the difference between a variable and a constant.

Solution 2.1

Method: Declaring, choosing and repeating

Worked steps

a variable's value can change while the program runs M1; a constant's value is fixed A1.

Practice 2.2 ▪ 1 mark

Give the value of `23 MOD 4`.

Solution 2.2

Worked steps

3 B1.

Practice 2.3 ▪ 2 marks

Give the pseudocode declaration for a variable that stores a student's name, and a constant for a tax rate of 0.2.

Solution 2.3

Method: Declaring, choosing and repeating

Worked steps

`DECLARE Name : STRING` B1; `CONSTANT TaxRate = 0.2` B1.

Practice 2.4 ▪ 3 marks

Identify the data type you would use for each of: a price, whether a box is ticked, a single grade letter.

Solution 2.4

Worked steps

Price: `REAL` B1; whether a box is ticked: `BOOLEAN` B1; a single grade letter: `CHAR` B1.

Practice 2.5 ▪ 2 marks

Explain the difference between a WHILE loop and a REPEAT loop.

Solution 2.5

Method: Declaring, choosing and repeating

Worked steps

A WHILE loop tests the condition before the body, so the body may run zero times B1; a REPEAT loop tests after the body, so it always runs at least once B1.

Exam-style 3.1 ▪ 1 mark

A loop that always runs **at least once** is a:

A FOR loop

B WHILE (pre-condition) loop

C REPEAT (post-condition) loop

D CASE statement

Solution 3.1

A FOR loop

B WHILE (pre-condition) loop

C REPEAT (post-condition) loop ✓

D CASE statement

Worked steps

C. A REPEAT (post-condition) loop runs at least once.

Exam-style 3.2 ▪ 1 mark

A program reads 10 marks and outputs their total.
(a) State which control structure repeats the input 10 times.
(b) Write pseudocode using a count-controlled loop to total 10 input marks.

Solution 3.2

Method: Declaring, choosing and repeating

Worked steps

(a) iteration (a count-controlled / FOR loop) B1.

(b) `total ← 0` M1; `FOR i ← 1 TO 10` M1; `INPUT mark` then `total ← total + mark` M1; `NEXT i` then `OUTPUT total` A1.

Exam-style 3.3 ▪ 2 marks

State the difference between a procedure and a function.

Solution 3.3

Method: Declaring, choosing and repeating

Worked steps

a function returns a value to where it was called M1; a procedure does a task but returns no value A1.

Exam-style 3.4 ▪ 3 marks

A program records the times, in whole seconds, taken by 20 runners.
(a) Complete the pseudocode declarations for the total, one runner's time and the loop counter.
(b) Write pseudocode that inputs the 20 times, adds them to a total, and outputs the average.
(c) The program must also output the fastest time. Describe, in words, the extra steps needed inside the loop.
(d) A time of 137 seconds must be shown as minutes and seconds. Give the two expressions, using MOD and DIV, and state the values they produce.
(e) The conversion in part (d) is needed in three places in the program. Explain why writing it as a function is better than repeating the code, and state what a function does that a procedure does not.

Solution 3.4

Method: Declaring, choosing and repeating

Worked steps

(a) `DECLARE Total : INTEGER`, `DECLARE Time : INTEGER`, `DECLARE Count : INTEGER` **B3** *one each.*
(b) `Total ← 0` before the loop **B1**; `FOR Count ← 1 TO 20` with `NEXT Count` **B1** *correct count of 20*; `INPUT Time` inside the loop **B1**; `Total ← Total + Time` inside the loop **B1**; `Average ← Total / 20` after the loop **B1**; `OUTPUT Average` **B1**.
(c) Set a variable, say `Fastest`, to the first time (or to a very large value) before the loop **B1**; inside the loop compare each new time with `Fastest` **B1**; if the new time is smaller, replace `Fastest` with it, and output `Fastest` after the loop **B1**.

Solution 3.4

(d) `137 DIV 60` gives 2 minutes B1; `137 MOD 60` gives 17 seconds B1; so the time is displayed as 2 minutes 17 seconds B1.
(e) The code is written once and called three times, so the program is shorter and a correction only has to be made in one place B1; it is also easier to read and test B1. A function returns a value to the line that called it, so it can be used inside an expression, while a procedure does not return a value B1.