

Week 24

IGCSE Computer Science

Homework bundle for this week

本周作业合集

exercises.lol

Contents 目录

Topic 8 quiz	2
Exercise sheet 8.1	5
Past-paper questions 8.1	9
Exercise sheet 8.2	14
Past-paper questions 8.2	18
Exercise sheet 8.3	40
Past-paper questions 8.3	44

IGCSE Computer Science

Name: _____ Score: _____

1. What is the difference between a variable and a constant?

- ☐ a variable can change while the program runs; a constant is fixed
- ☐ a constant can change; a variable is fixed
- ☐ they are the same
- ☐ a variable cannot hold a number

2. What is '17 MOD 5'?

3. An array is:

- ☐ one variable that holds many values of the same type, found by an index
- ☐ a single value
- ☐ several variables of different types
- ☐ a file on disk

4. Which structure chooses which steps to run based on a condition?

- ☐ selection (IF / CASE)
- ☐ sequence
- ☐ iteration
- ☐ a procedure

5. What is '17 DIV 5'?

6. In 'grid[2, 3]', which cell is accessed?

- ☐ row 2, column 3
- ☐ row 3, column 2
- ☐ the 23rd element
- ☐ column 2 only

7. A WHILE loop tests its condition before the body (so it may run zero times), while a REPEAT...UNTIL loop tests after the body (so it always runs at least once).

- ☐ True ☐ False

8. What does 'LENGTH("Computer")' return?

9. You should always close a file when you have finished using it.

☐ True ☐ False

10. Which line keeps a running total?

☐ $\text{total} \leftarrow \text{total} + \text{value}$

☐ $\text{count} \leftarrow \text{count} + 1$

☐ $\text{total} \leftarrow 0$

☐ OUTPUT total

1. a variable can change while the program runs; a constant is fixed

A variable's value can change; a constant is set once and never changes.

2. 2

MOD gives the remainder: $17 \div 5 = 3$ remainder 2.

3. one variable that holds many values of the same type, found by an index

An array stores many same-type values under one name; each is accessed by its index.

4. selection (IF / CASE)

Selection branches on a condition; sequence runs in order; iteration repeats.

5. 3

DIV gives the whole-number part: $17 \div 5 = 3$ (remainder discarded).

6. row 2, column 3

The first index is the row, the second the column — row 2, column 3.

7. True

Pre-condition (WHILE) can skip entirely; post-condition (REPEAT) always runs once — choose by whether the body must run at least once.

8. 8

"Computer" has 8 characters.

9. True

Closing saves any buffered data and frees the file for other programs.

10. $\text{total} \leftarrow \text{total} + \text{value}$

' $\text{total} \leftarrow \text{total} + \text{value}$ ' accumulates the sum; ' $\text{count} \leftarrow \text{count} + 1$ ' counts occurrences.

8.1 Programming concepts

Name: _____ Class: _____ Date: _____

Total: 11 marks

KEY WORDS variable 变量 • output 输出 • input 输入 • function 函数 • procedure 过程

Objective

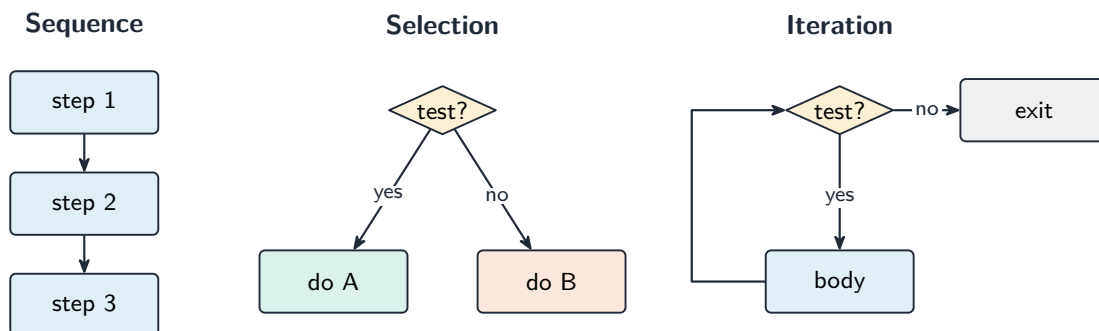
Build the skills to answer exam questions on **programming concepts** 编程概念 — variables, data types, the three control structures (**sequence**, **selection**, **iteration** 顺序、选择、迭代), operators, and procedures/functions.

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

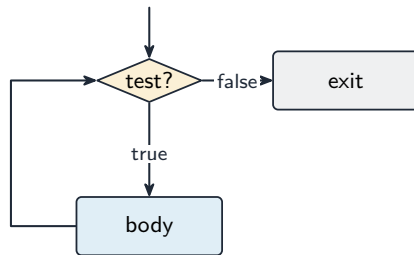
1 Worked examples

■ Control structures



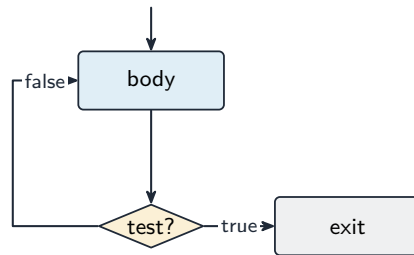
Sequence runs steps in order; selection chooses a branch; iteration repeats a body

WHILE (pre-condition)



tested first $\Rightarrow$ may
run **zero** times

REPEAT (post-condition)



tested last $\Rightarrow$ runs
at least once

A WHILE loop may run 0 times; a REPEAT loop runs at least once

- $17 \text{ MOD } 5 = 2$ (remainder); $17 \text{ DIV } 5 = 3$ (whole part).

2 Practice

2.1 State the difference between a variable and a constant.

[2]

2.2 Give the value of $23 \text{ MOD } 4$.

[1]

3 Exam-style questions

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

[1]

- A FOR loop
- B WHILE (pre-condition) loop
- C REPEAT (post-condition) loop
- D CASE statement

3.2 A program reads 10 marks and outputs their total.

(a) State which control structure repeats the input 10 times.

[1]

(b) Write pseudocode using a count-controlled loop to total 10 input marks. [4]

3.3 State the difference between a procedure and a function. [2]

Solutions

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

2.2 3.

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

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

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

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

- 3** Tick (✓) **one** box to identify the operator that returns the quotient of a calculation with the fractional part discarded.

A / ☐

B DIV ☐

C ^ ☐

D MOD ☐

[1]

- 1** Tick (✓) **one** box to complete this sentence.

The result of the arithmetic operation $4 \wedge 2$ is

A 2 ☐

B 8 ☐

C 16 ☐

D 42 ☐

[1]

- 3** Tick (✓) **one** box to identify the most appropriate data type to store 'M'

A integer ☐

B char ☐

C real ☐

D Boolean ☐

[1]

5 Two library routines used in programming are MOD and DIV.

State the purpose of each of the library routines.
Give **one** example pseudocode statement for each of the library routines.

MOD purpose

.....

Example pseudocode

.....

DIV purpose

.....

Example pseudocode

.....

[4]

2 Tick (✓) **one** box to complete this sentence.

A pseudocode example of a selection statement is

- A

CALL Sorting(Value1, Value2)

☐
- B

DECLARE Count : INTEGER

☐
- C

IF X = 7 THEN Y ← 21 ENDIF

☐
- D

WHILE X <> -1 DO

☐

[1]

3 State what is meant by the data types integer and real.
Give an example of each.

Integer

.....

Example

Real

.....

Example

[4]

5 A high-level programming language makes use of arithmetic, Boolean and logical operators.

State how each type of operator is used.
Give an example statement, in pseudocode, for each one.

Arithmetic

.....

Example

.....

Boolean

.....

Example

.....

Logical

.....

Example

.....

[6]

3 Describe the characteristics of the string and char data types and give an example of each.

String

.....

Example

Char

.....

Example

[4]

8.2 Arrays

Name: _____ Class: _____ Date: _____

Total: 9 marks

KEY WORDS two-dimensional (2d) 二维 • one-dimensional (1d) 一维 • arrays 数组 • index 索引
• output 输出

Objective

Build the skills to answer exam questions on **arrays** 数组 — declaring and using **one-dimensional (1D)** 一维 and **two-dimensional (2D)** 二维 arrays, and using a loop to fill or read them.

You must be able to:

- declare a 1D and a 2D array and use an **index** 索引 to read/write a value
- use a loop (and a nested loop) to process an array
- choose between a 1D and a 2D array for a problem

1 Worked examples

■ 1D and 2D arrays

index	1	2	3	4	5
scores	90	75	60	88	50

↑
scores[2] is 75

A 1D array is a list; each value is found by one index

		column		
		1	2	3
row	1			
	2			
	3			

← grid[2,3]

A 2D array is a table; each value is found by two indexes [row, column]

- 1D: DECLARE scores : ARRAY[1:5] OF INTEGER, then scores[2] ← 75.
- 2D: DECLARE grid : ARRAY[1:3, 1:3] OF INTEGER, then grid[2, 3] ← 8.

2 Practice

2.1 An array is declared DECLARE temps : ARRAY[1:7] OF REAL. State how many values it can hold. [1]

2.2 State one difference between a 1D and a 2D array. [2]

3 Exam-style questions

3.1 Which data structure best stores the marks of 30 students in **one** subject? [1]

- **A** a single variable
- **B** a 1D array
- **C** a 2D array
- **D** a constant

3.2 An array names holds 5 names in elements 1 to 5.

(a) Write pseudocode to output every name using a loop. [3]

(b) State which type of array (1D or 2D) you would use to store the marks of 30 students in **5** subjects, and why. [2]

Solutions

2.1 7.

2.2 a 1D array is a single list using one index; a 2D array is a table using two indexes [row, column].

3.1 B. One subject = a single list = a 1D array.

3.2 (a) `FOR i ← 1 TO 5; OUTPUT names[i]; NEXT i.`

(b) a 2D array; because each student needs 5 marks, so two indexes [student, subject] are required.

11 The scores for each competitor in an archery competition are recorded.

The one-dimensional (1D) array `CompetitorName[]` is used to store the names of the competitors.

The maximum number of competitors is 30.

There are 10 rounds in the competition with a maximum score of 30 per round.

The two-dimensional (2D) array `CompetitorScore[]` stores the score in each of the 10 rounds for each competitor.

The first index of the array `CompetitorScore[]` is the same as the index of the competitor in `CompetitorName[]`

After the 10 rounds have been completed, the highest score and the lowest score for each competitor are discarded.

The overall score for each competitor is the total of the remaining 8 scores (after discarding the highest and lowest scores) .

A competitor with an overall score of 210 or more moves onto the next competition. A competitor with an overall score between 180 and 209 inclusive becomes a reserve competitor. A competitor with an overall score below 180 does not qualify.

Write a program that meets the following requirements:

- inputs the total score (out of 30) for each competitor in each round
- validates each input is between 0 and 30 inclusive
- calculates the overall score for each competitor discarding the highest and lowest scores
- outputs the name of each competitor and if they move onto the next competition, are a reserve or do not qualify
- calculates and outputs the number of competitors who:
 - have qualified for the next competition
 - are reserve competitors
 - do **not** qualify.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

..... [15]

- 2 This pseudocode algorithm is intended to input a first name from a user and search for it in a two-dimensional (2D) array. If the name is found, the algorithm outputs all the remaining details for that name. The algorithm will continue to allow other names to be entered for searching, until the user input stops it.

```
01 DECLARE Contacts : ARRAY[1:500, 1:4] OF REAL
02 DECLARE Row : INTEGER
03 DECLARE Column : INTEGER
04 DECLARE Continue : BOOLEAN
05 DECLARE Stop : BOOLEAN
06 DECLARE FirstName : STRING
07 Continue ← TRUE
08 Stop ← FALSE
09 WHILE Continue
10     Row ← 1
11     OUTPUT "Enter a first name "
12     OUTPUT FirstName
13     REPEAT
14         IF Contacts[Row, 1] = FirstName
15             THEN
16                 FOR Column ← 1 TO 4
17                     OUTPUT Contacts[Row, 1]
18                 NEXT Column
19                 Stop ← TRUE
20             ELSE
21                 Row ← Row + 1
22             ENDIF
23     NEXT Stop OR Row > 500
24     OUTPUT "Search for another name? (Y or N) "
25     INPUT Answer
26     IF Answer = 'N' OR Answer = 'n'
27         THEN
28             Continue ← TRUE
29     ENDIF
30 ENDWHILE
```

- (a) Identify the line numbers of **five** errors in the pseudocode and suggest a correction for each error.

Error 1 line number

Correction

.....

Error 2 line number

Correction

.....

Error 3 line number

Correction

.....

Error 4 line number

Correction

.....

Error 5 line number

Correction

.....

[5]

(b) A procedure `NewData` allows data for new contacts to be entered and stored in the 2D array `Contacts[]`. A parameter, `Number`, is used to pass values from the main algorithm to the procedure for the number of new contacts. Next, four pieces of contact data for each new contact are entered and stored at the start of the array.

(i) Complete the pseudocode for `PROCEDURE NewData`

PROCEDURE `NewData` (.....)

FOR `Row` ← 1 TO

FOR `Column` ← 1 TO 4

INPUT

.....

.....

ENDPROCEDURE

[4]

(ii) Write the pseudocode to input the number of new contacts and to use procedure `NewData`. A variable declaration is **not** required.

.....

.....

.....

(c) A first name is used as the search key in the algorithm on page 2. If the name is found, all the data for that name is output and the algorithm stops searching.

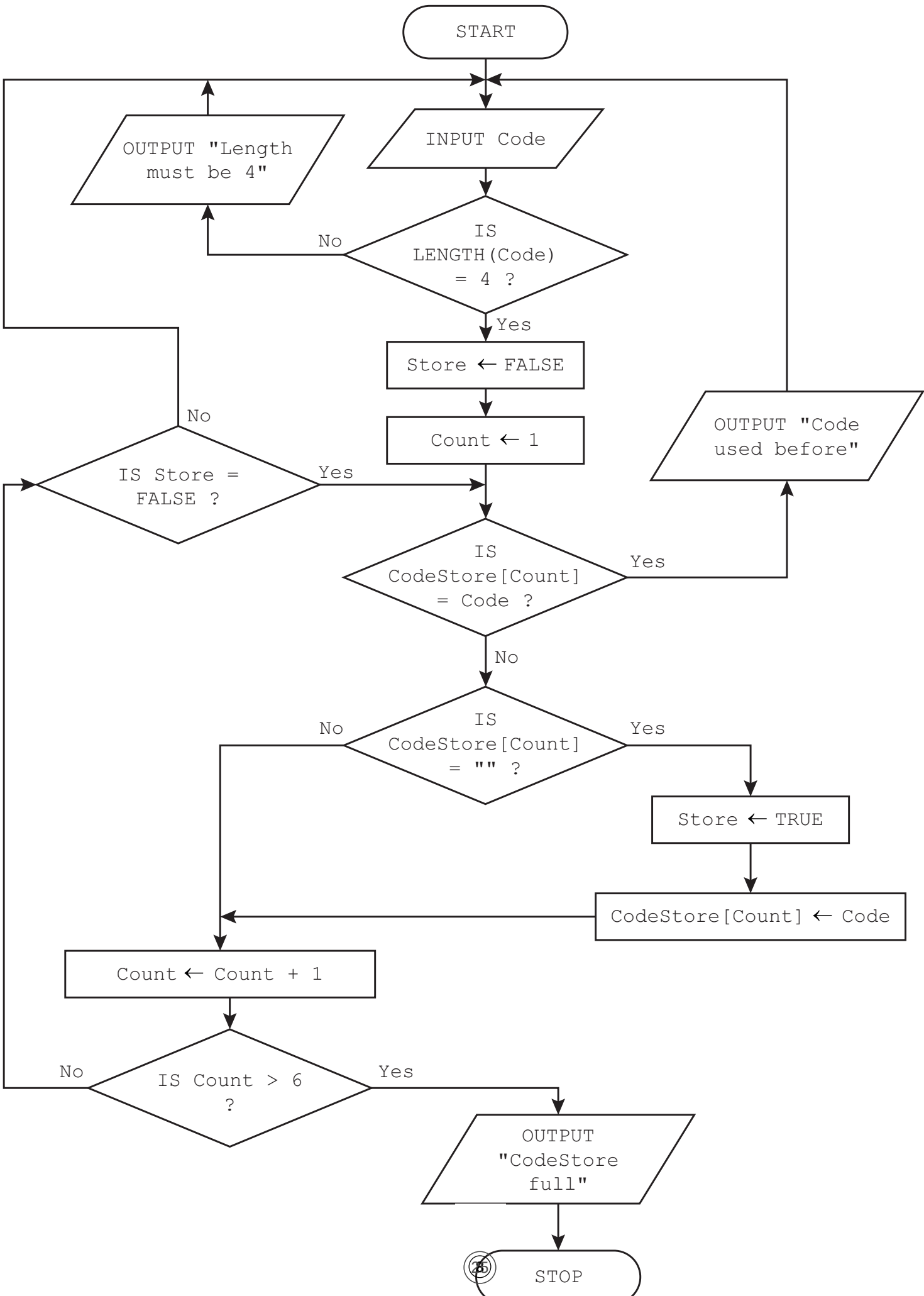
(i) Explain why the given algorithm may **not** always give the expected results.

[2]

(ii) Explain how you could improve the algorithm so that the search is more likely to find the required results.

[2]

5 This flowchart represents an algorithm.



(a) The first data row of the trace table shows the values currently stored in `CodeStore[]`

Complete the trace table for the algorithm, using the input data:

7686, Face, Speed, 432U, Yp79

[illegible]

[6]

(b) Describe the processes in the algorithm on page 6.

..... [3]

(c) The algorithm on page 6 requires the array `CodeStore[]` to be declared as a one-dimensional (1D) string array with six elements that are initially set to the null string `""`.

Write the pseudocode required to declare and initialise the array `CodeStore[]`.
You do need to declare any variables used.

..... [4]

- them in the array `Names [ ]`

```
FOR Count  $\leftarrow$  1 TO 10
```

```
OUTPUT "Please enter a name"
```

INPUT Name

```
Names[Count] ← Name
```

NEXT Count

Rewrite the algorithm in pseudocode, using a condition-controlled loop instead of a count-controlled loop.

[4]

3 The purpose of this pseudocode algorithm is to carry out a bubble sort to sort, in descending order, 1000 numbers stored in a one-dimensional (1D) array.

```
01 DECLARE Values : ARRAY[1:1000] OF REAL
02 DECLARE Index : CHAR
03 DECLARE Stop : BOOLEAN
04 DECLARE Hold : REAL
05 Stop ← FALSE
06 WHILE NOT Stop DO
07     Stop ← FALSE
08     FOR Index ← 1 TO 50
09         IF Values[Index + 1] > Values[Index]
10             THEN
11                 Hold ← Values[Index]
12                 Values[Index] ← Values[Index + 1]
13                 Values[Index + 1] ← Hold
14                 Stop ← FALSE
15             ENDIF
16     NEXT Index
17 NEXT Stop
```

(a) Identify the line numbers of **four** errors in the pseudocode and suggest a correction for each error.

Error 1 line number

Correction
.....

Error 2 line number

Correction
.....

Error 3 line number

Correction
.....

Error 4 line number

Correction
.....

(b) The swap section in lines 11 to 13 of the existing code are to be changed to a call statement for a procedure. This procedure will include two parameters representing the indexes of the array elements to be swapped.

(i) Complete the pseudocode for PROCEDURE Swap

```
PROCEDURE Swap ( ..... )  
  
    DECLARE Hold : REAL  
  
    Hold ← Values[ ..... ]  
  
    .....  
  
    .....  
  
ENDPROCEDURE
```

[4]

(ii) Write the pseudocode to transfer control to PROCEDURE Swap

```
.....  
  
.....
```

[2]

(c) Explain the difference between global variables and the variable declared in 3(b)(i).

```
.....  
  
.....  
  
.....  
  
.....
```

[2]

- 12** A one-dimensional (1D) array `Rooms[]` contains the names of up to 20 rooms in a house. A two-dimensional (2D) array `Dimensions[]` is used to store the length, width and area of each room.

The position of any room's data is the same in both arrays. For example, the data in index 5 of `Dimensions[]` belongs to the room in index 5 of `Rooms[]`

The variable `Number` stores the number of rooms for which data is to be input. There must be at least 3 rooms but no more than 20.

Write a program that meets the following requirements:

- allows the number of rooms for which data is required to be input, stored and validated
- allows the name of the room and the length and width of the room, in metres, to be entered and stored
- allows the area of each room to be calculated as length multiplied by width and stored as square metres rounded to two decimal places
- calculates the average size of all the rooms by area, in square metres, rounded to two decimal places
- finds the largest room and smallest room by area
- outputs the names of all rooms with their dimensions and area
- outputs the names of the largest room and smallest room by area
- outputs the total area of the house and the average size of all the rooms by area.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

This image shows a full page of white paper designed for handwriting practice. It features ten evenly spaced, horizontal dashed lines that run across the entire width of the page. There are no margins, text, or other markings present.

[15]

10 The one-dimensional (1D) array `Clubs[]` is used to store the names of 12 cricket clubs in a local sports league.

The two-dimensional (2D) array `Statistics[]` is used to store, for each cricket club, the number of:

- matches won
- matches drawn
- matches lost.

The 1D array `Points[]` is used to store the total number of points each cricket club has been awarded.

The position of any cricket club's data is the same in all three arrays. For example, the data in index 2 of `Statistics[]` and index 2 of `Points[]` belongs to the cricket club in index 2 of `Clubs[]`

The variable `Matches` stores the number of matches played by each team. Each team plays the same number of matches.

Points are awarded for:

- a win – 12 points
- a draw – 5 points
- a loss – 0 points.

Write a program that meets the following requirements:

- allows the number of matches played to be input and stored, with a maximum of 22 matches
- validates the number of matches played
- allows the names of the cricket clubs to be input and stored
- allows the number of matches won, drawn and lost to be input and stored for each team
- validates the number of matches won, drawn or lost against the number of matches played
- asks the user to re-enter the number of matches won, drawn or lost if the total does not match the number of matches played
- calculates and stores the total number of points for each club
- finds the cricket club or clubs with the highest number of points
- outputs the name or names of the winning club or clubs, the number of wins and the total number of points awarded.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

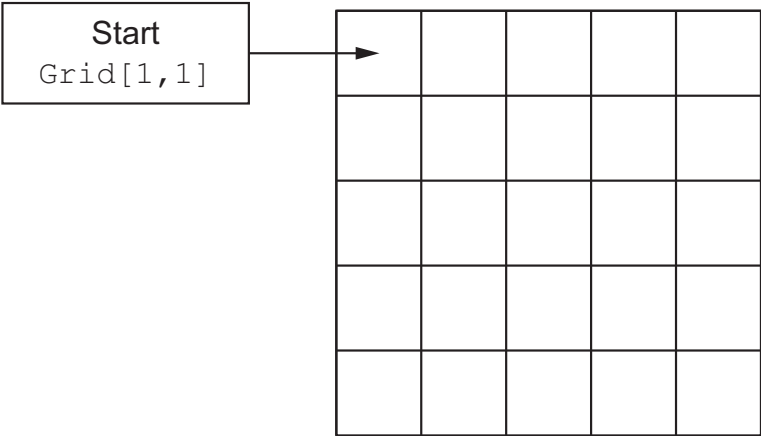
All inputs and outputs must contain suitable messages.

[illegible]

[illegible]

[illegible]

11 A one-player game uses the two-dimensional (2D) array `Grid[]` to store the location of a secret cell to be found by the player in 10 moves. Each row and column has 5 cells.



At the start of the game:

- The program places an 'X' in a random cell (**not** in `Grid[1,1]`) and empties all the other cells in the grid.
- The player starts at the top left of the grid.
- The player has 10 moves.

During the game:

- The player can move left, right, up or down by one cell and the move must be within the grid.
- "You Win" is displayed if the player moves to the cell with 'X' and has played 10 moves or less.
- "You Lose" is displayed if the player has made 10 moves without finding the 'X'.

Write a program that meets these requirements.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.





22

8.3 File handling

Name: _____ Class: _____ Date: _____

Total: 9 marks

KEY WORDS file 文件 • file handling 文件处理 • variable 变量 • array 数组 • constant 常量

Objective

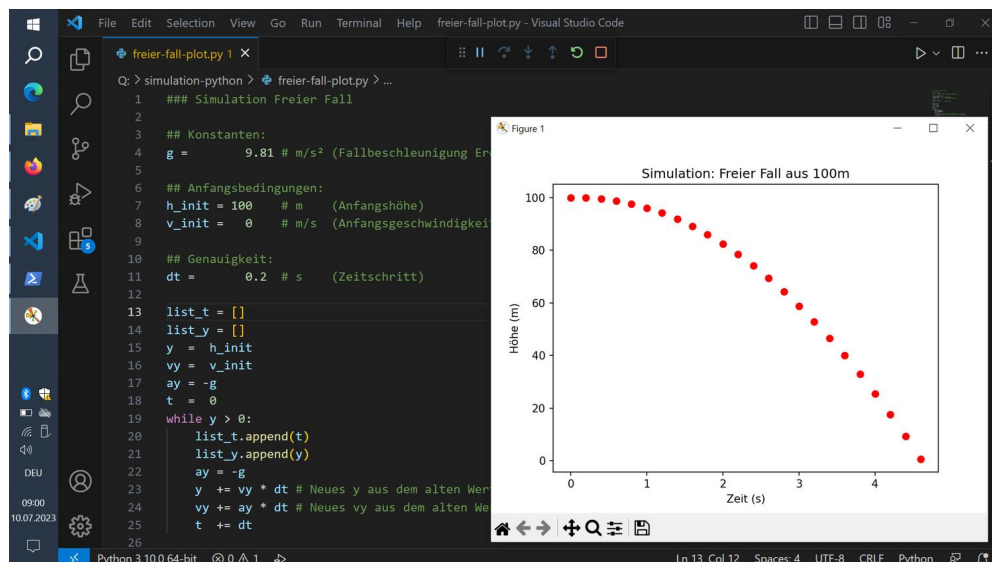
Build the skills to answer exam questions on **file handling** 文件处理 — storing data in a **file** 文件 so it is kept after the program stops, and the **open** → **use** → **close** pattern for reading and writing.

You must be able to:

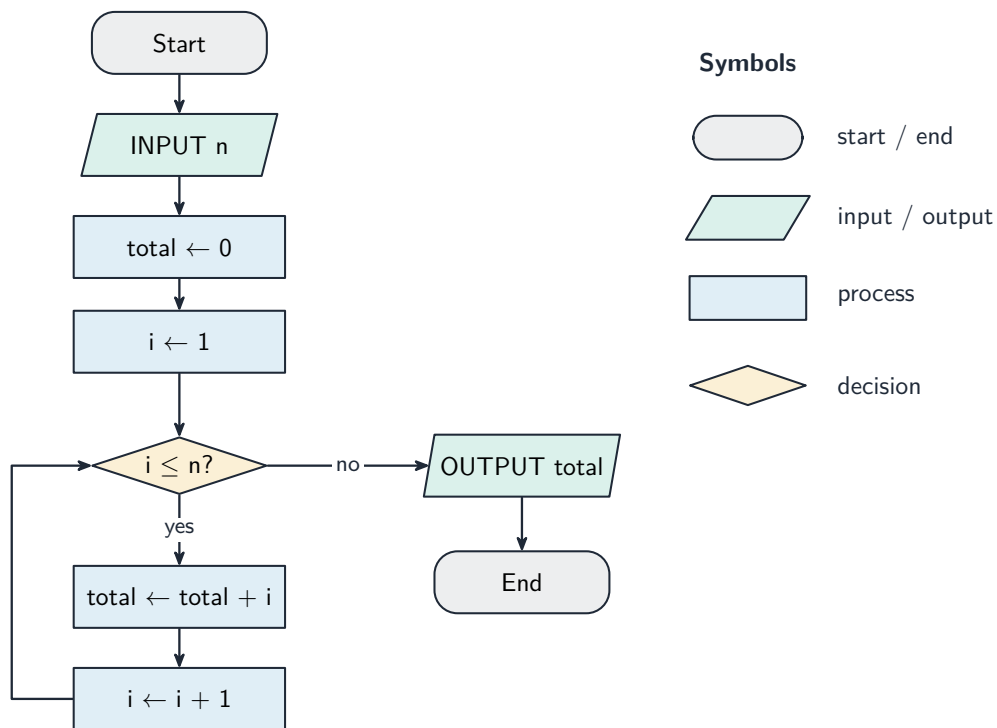
- explain why a program uses a file to store data
- write pseudocode to **open, write, read and close** a file
- describe the difference between writing to and reading from a file

1 Worked examples

■ Open, use, close



Data written to a file is kept after the program stops, unlike a variable in memory



File handling follows an open - read/write - close sequence

Write then read back one line:

- `OPENFILE "data.txt" FOR WRITE → WRITEFILE "data.txt", "Hello" → CLOSEFILE "data.txt"`
- `OPENFILE "data.txt" FOR READ → READFILE "data.txt", line → CLOSEFILE "data.txt"`

Always **close** a file when finished.

2 Practice

2.1 State one reason a program stores data in a file rather than in a variable. [1]

2.2 State the three steps used to handle a file, in order. [3]

3 Exam-style questions

3.1 A program must keep its data after it stops running. The data should be stored: [1]

- **A** in a variable
 - **B** in a constant
 - **C** in a file
 - **D** in an array
-

3.2 A program writes a player's name to a file, then later reads it back.

(a) Write pseudocode to open `scores.txt`, write the name in variable `playerName` to it, and close it. [3]

(b) State what could go wrong if the file is **not** closed after writing. [1]

Solutions

2.1 a file keeps the data after the program stops / it is permanent (non-volatile) storage.

2.2 open the file; read or write the data; close the file.

3.1 C. A file keeps data after the program stops.

3.2 (a) `OPENFILE "scores.txt" FOR WRITE; WRITEFILE "scores.txt", playerName;`
`CLOSEFILE "scores.txt".`

(b) the data may not be saved properly / the file may be left locked or corrupted.

9 A school stores all the students' attendance data in a file called `Attendance.txt`

Write an algorithm in pseudocode to read a line of text from the file and store this line in the variable `StudentAttendance`

[4]

10 The variables `Seconds` and `Minutes` are used in a program. `Seconds` holds the number of seconds as a whole number and `Minutes` holds the number of minutes as a whole number.

(a) Write pseudocode statements to declare the variables `Seconds` and `Minutes`

[1]

(b) The procedure `Time(TotalSeconds)` takes the number of seconds as a parameter. The procedure converts the value held in the parameter into minutes and seconds and outputs the result.

Write this procedure in pseudocode.

[4]

2 Tick (✓) **one** box to complete this sentence.

The pseudocode to store a hotel name held in the variable `Name` to a text file is

A `READFILE Hotels.txt, Name`

☐

B `WRITEFILE Hotels.txt, Name`

☐

C `WRITEFILE Name, Hotels.txt`

☐

D `STOREFILE Hotels.txt, Name`

☐

[1]