A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

11.2.2 Count-controlled loops: FOR ... NEXT, totals and counts

Name: _____ Class: _____ Date: _____ **Total: 42 marks**

KEY WORDS count-controlled loop 计数循环 • iteration 迭代 • running total 累计总和 • nested loops 嵌套循环 • counter 计数器

Objective

Learn to write a **count-controlled loop** 计数循环: the `FOR … NEXT` loop, which repeats a group of statements a **known number of times**. You use it to find a total, a count, a mean and a largest value, and you put one loop inside another. Selection was on sheets 11.2 and 11.2.1; the other two loops come on sheet 11.2.3.

You must be able to:

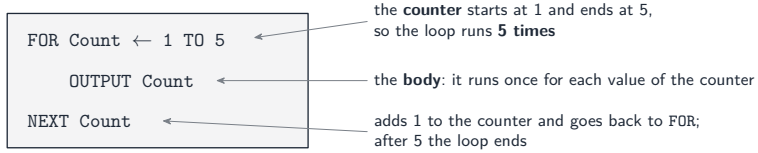
- write a `FOR … NEXT` loop, with and without `STEP`
- state how many times a loop runs, and what it outputs
- use a loop to find a **running total** 累计总和, a **count**, a mean, and the largest or smallest value
- put an `IF` statement inside a loop
- write and follow **nested loops** 嵌套循环

1 Worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

■ The FOR ... NEXT loop

Iteration 迭代 means repeating statements. When the number of repetitions is **known before the loop starts**, use a count-controlled loop.



- The **counter** (here `Count`) is an `INTEGER` variable. The loop gives it each value from the start value to the end value, one after the other.

- `FOR Count ← 1 TO 5` runs **5** times, and `FOR Count ← 0 TO 5` runs **6** times. In general the number of passes is *end − start + 1*.
- Write the counter after `NEXT`. It shows which loop ends there.

`STEP` changes the counter by a different amount:

```
FOR Number ← 10 TO 0 STEP -2
    OUTPUT Number
NEXT Number
```

This outputs 10, 8, 6, 4, 2, 0. Without `STEP`, the counter goes up by 1.

■ Loop patterns: a total and a count

Input the temperature at noon on each of 30 days. Output how many days were below zero, and the mean temperature.

```
DECLARE Day, BelowZero : INTEGER
DECLARE Temp, Total : REAL
Total ← 0
BelowZero ← 0
FOR Day ← 1 TO 30
    INPUT Temp
    Total ← Total + Temp
    IF Temp < 0 THEN
        BelowZero ← BelowZero + 1
    ENDIF
NEXT Day
OUTPUT BelowZero
OUTPUT Total / 30
```

- **Running total:** set it to 0 **before** the loop, and add to it **inside** the loop.
- **Count:** set it to 0 before the loop, and add 1 inside an `IF`.
- **Output after the loop.** An `OUTPUT` inside the loop would run 30 times.
- If `Total ← 0` were inside the loop, the total would go back to 0 on every pass. This is a very common error.

■ Loop pattern: the largest value

Output the highest of the 30 temperatures.

```
INPUT Highest
FOR Day ← 2 TO 30
    INPUT Temp
    IF Temp > Highest THEN
        Highest ← Temp
    ENDIF
NEXT Day
OUTPUT Highest
```

- Keep the **largest value so far** in a variable. Each new value replaces it only if it is greater.
- Start with the **first value**, not with 0: every temperature could be below zero. The loop then runs from **2**.
- For the smallest value, change `>` to `<`.

Nested loops

Input `Rows`*, and output a triangle of stars: 1 star on the first row, 2 on the second, and so on.*

```
INPUT Rows
FOR Row ← 1 TO Rows
    Line ← ""
    FOR Star ← 1 TO Row
        Line ← Line & "*"
    NEXT Star
    OUTPUT Line
NEXT Row
```

- The **inner** loop runs completely for **each** pass of the **outer** loop.
- The two loops need **different** counters, and the inner loop must end (`NEXT Star`) before the outer loop ends (`NEXT Row`).
- When `Rows` is 3, the statement inside the inner loop runs $1 + 2 + 3 = 6$ times.

2 Practice

Now use the methods above. The first questions need one idea each.

2.1 State how many times each loop runs. [3]

(a) `FOR Index ← 1 TO 20`

(b) `FOR Index ← 0 TO 9`

(c) `FOR Index ← 5 TO 25 STEP 5`

2.2 State what each loop outputs. [2]

(a)

```
FOR Count ← 3 TO 6
    OUTPUT Count * 2
NEXT Count
```

(b)

```
FOR Count ← 9 TO 1 STEP -3
    OUTPUT Count
NEXT Count
```

2.3 Write a `FOR` loop that outputs the **even** numbers from 2 to 20. [2]

2.4 Complete the pseudocode so that it inputs 10 values and outputs their mean. Write what goes in each blank. [4]

```
Total ← ..........(1)
FOR Index ← 1 TO 10
    INPUT Value
    Total ← ..........(2)
NEXT ..........(3)
OUTPUT ..........(4)
```

2.5 A student wants to add up five prices, but the output is always the last price that was input.

```
FOR Item ← 1 TO 5
    Total ← 0
    INPUT Price
    Total ← Total + Price
NEXT Item
OUTPUT Total
```

Explain the error, and state how to correct it. [2]

2.6 Write pseudocode that inputs the heights of 25 students, one at a time, and outputs the smallest height. [4]

3 Exam-style questions

3.1 Write pseudocode that inputs an integer `Table`, and then outputs the first 12 lines of its multiplication table. For the input 7, the first two lines are:

```
1 x 7 = 7
2 x 7 = 14
```

Declare your variables. [5]

3.2 A swimming club times 40 swimmers over 100 metres. An algorithm will:

- input the time of each swimmer, in seconds, one value at a time
- count the swimmers with a time of less than 60 seconds
- output this count, and the mean time of all the swimmers.

Write pseudocode for the algorithm. Declare your variables. [6]

3.3 A weather station records the rainfall, in mm, for each of the 30 days of a month. An algorithm will:

- input the rainfall for each day, one value at a time
- count the days with no rain
- find the total rainfall for the month
- find the day with the most rain; if two days have the same rainfall, keep the earlier day
- output the count, the total and the day number.

Write pseudocode for the algorithm. Declare any variables you use. [7]

3.4 Study this pseudocode.

```
FOR Outer ← 1 TO 3
    FOR Inner ← 1 TO 4
        OUTPUT Outer * Inner
    NEXT Inner
NEXT Outer
```

(a) State how many values are output. [1]

(b) State the first **five** values that are output. [2]

(c) The programmer wants each pass of the **outer** loop to produce one line of output, for example `1 2 3 4`. Rewrite the pseudocode. Use a string variable `Line`, and the function `NUM_TO_STR()` to change a number into a string. [4]

Solutions

Each answer shows the steps, then the **result**.

2.1

(a) $20 - 1 + 1 =$ **20** times

(b) $9 - 0 + 1 =$ **10** times

(c) the counter takes the values 5, 10, 15, 20, 25: **5** times

2.2

(a) the counter is 3, 4, 5, 6, and each is doubled: **6, 8, 10, 12**

(b) the counter goes down by 3 each time: **9, 6, 3**

2.3

- `FOR … STEP 2` from 2 to 20

```
FOR i ← 2 TO 20 STEP 2
    OUTPUT i
NEXT i
```

- *(outputting `i` only when `i MOD 2 = 0` also scores)*

2.4

- (1) the total starts at `0`
- (2) each value is added: **`Total + Value`**
- (3) the counter of this loop: **`Index`**
- (4) the mean of 10 values: **`Total / 10`**

2.5

- `Total ← 0` is **inside** the loop, so the total goes back to 0 on every pass, and only the last price is left
- move `Total ← 0` to the line **before** `FOR`

2.6

- input the first height and store it as the smallest
- a loop that runs for the other 24 heights
- compare each height with the smallest, and replace it if smaller
- output the smallest after the loop


```
DECLARE Count : INTEGER
DECLARE Height, Smallest : REAL
INPUT Height
Smallest ← Height
FOR Count ← 2 TO 25
    INPUT Height
    IF Height < Smallest THEN
        Smallest ← Height
    ENDIF
NEXT Count
OUTPUT Smallest
```

3.1

- the variables declared, and `INPUT Table`
- a `FOR` loop from 1 to 12, closed with `NEXT`
- the product `Line * Table` worked out inside the loop
- the `OUTPUT` has the values and the text in the correct order
- the text `" x "` and `" = "` is in quotes, with commas between the items

```
DECLARE Table, Line : INTEGER
INPUT Table
FOR Line ← 1 TO 12
    OUTPUT Line, " x ", Table, " = ", Line * Table
NEXT Line
```

3.2

- the variables declared with suitable types
- the total and the count set to 0 before the loop
- a `FOR` loop that runs 40 times, with the input inside it
- each time added to the total
- an `IF` that adds 1 to the count when the time is less than 60
- the count and `Total / 40` output **after** the loop

```
DECLARE Swimmer, Fast : INTEGER
DECLARE Time, Total : REAL
Total ← 0
Fast ← 0
FOR Swimmer ← 1 TO 40
    INPUT Time
    Total ← Total + Time
    IF Time < 60 THEN
        Fast ← Fast + 1
    ENDIF
NEXT Swimmer
OUTPUT Fast
OUTPUT Total / 40
```

3.3

- declarations; a count-controlled loop from 1 to 30; input inside the loop
- count the days with rainfall 0; running total
- keep the largest rainfall and its day number, replacing only when greater
- output all three after the loop

```
DECLARE Day, DryDays, WettestDay : INTEGER
DECLARE Rain, Total, MostRain : REAL
DryDays ← 0
Total ← 0
MostRain ← -1
WettestDay ← 0
FOR Day ← 1 TO 30
    INPUT Rain
    IF Rain = 0 THEN
        DryDays ← DryDays + 1
    ENDIF
    Total ← Total + Rain
    IF Rain > MostRain THEN
        MostRain ← Rain
        WettestDay ← Day
    ENDIF
NEXT Day
OUTPUT DryDays
OUTPUT Total
OUTPUT WettestDay
```

- `MostRain` starts at -1, below any real rainfall, so day 1 is always stored first; `>` rather than `>=` keeps the earlier day

3.4

(a) the inner loop runs 4 times for each of the 3 passes of the outer loop: $3 \times 4 = \mathbf{12}$ values

(b) the first pass of the outer loop gives 1×1 to 1×4, and the second pass starts with 2×1

- **1, 2, 3, 4, 2**

(c) `Line` is set to an empty string at the start of **each** pass of the outer loop

- inside the inner loop, the product is changed to a string and joined to `Line` with `&`
- a space is joined between the values
- `OUTPUT Line` comes after `NEXT Inner` and before `NEXT Outer`


```
FOR Outer ← 1 TO 3
    Line ← ""
    FOR Inner ← 1 TO 4
        Line ← Line & NUM_TO_STR(Outer * Inner) & " "
    NEXT Inner
    OUTPUT Line
NEXT Outer
```