

Exercise walk-through

11.2.1

CASE structures, and choosing between IF and CASE

A-Level Computer Science

The post-condition loop, and choosing the right loop

post-condition loop : tests *after* each pass, so it always runs *at least once*

REPEAT

INPUT password

UNTIL password = correctPassword

If ...

use

the count is known

FOR

it may run zero times

WHILE

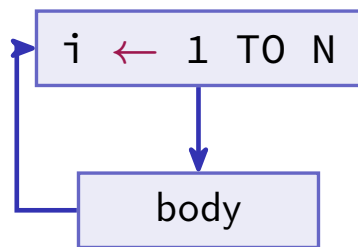
it must run at least once

REPEAT

Justify the choice by whether the count is *known* and whether the body must run *at least once*.

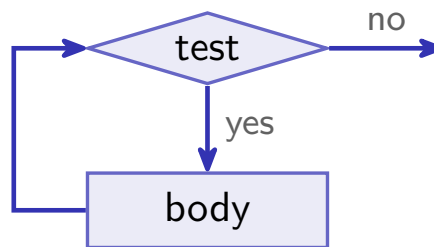
Where each loop tests its condition

FOR



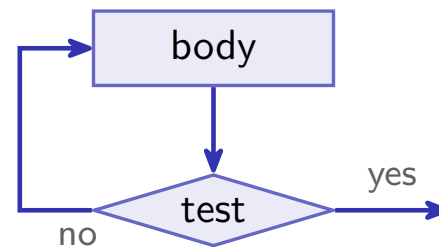
a set number
of passes

WHILE



test **before** the body:
may run 0 times

REPEAT



test **after** the body:
runs at least once

The three loops differ only in **where the condition is tested**: before the body, after it, or a set number of times.

Two loops you must know: validation and a rogue value

Validation: ask until the value is valid

```
REPEAT
  OUTPUT "Enter a mark, 0 to 100"
  INPUT Mark
UNTIL Mark >= 0 AND Mark <= 100
```

The mark must be input **at least once** before it can be tested: post-condition.

Rogue value: stop at -1 , and do not use it

```
Total ← 0
INPUT Price
WHILE Price <> -1
  Total ← Total + Price
  INPUT Price
ENDWHILE
```

The first input could be -1 , so the loop may run **zero** times: pre-condition.

Justifying the loop: the two marks

One mark for the **name** of the loop and one for the **reason**, in the scheme's words:

count-controlled

because the number of iterations is known before the loop starts

post-condition

because the loop body must be executed at least once

pre-condition

because the loop may not need to execute at all

A loop over the four elements of an array written as a **WHILE** with a counter is “not the most appropriate”: the count, four, is known, so a **FOR** loop fits.

Worked example: which loop suits each task?

Ask **how many times** the body runs, and **when the test happens**.

(a) print the 12 times table

FOR

the count, 12, is known in advance

(b) read numbers until the user enters 0

WHILE

the first input might already be 0, so test before the body: zero or more times

(c) ask for a password until it is correct

REPEAT ... UNTIL

you must ask at least once before there is anything to test: one or more times

The deciding question: must the body run at least once? **WHILE** may run zero times; **REPEAT** always runs once.

Exercise 2.1 (a)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(a) Day is 2

[1]

Exercise 2.1 (a)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(a) Day is 2

[1]

(a) the value 2 matches the second line: **“Tuesday”**

Exercise 2.1 (b)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(b) Day is 7

[1]

Exercise 2.1 (b)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(b) Day is 7

[1]

(b) 7 is in the range 6 to 7: **“Weekend”**

Exercise 2.1 (c)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(c) Day is 4

[1]

Exercise 2.1 (c)

2.1 Study this pseudocode.

```
CASE OF Day
  1 : OUTPUT "Monday"
  2 : OUTPUT "Tuesday"
  6 TO 7 : OUTPUT "Weekend"
  OTHERWISE : OUTPUT "Midweek"
ENDCASE
```

M	T	W	T	F	S	S

State what is output when:

(c) Day is 4

[1]

(c) 4 matches no line, so OTHERWISE runs: "Midweek"

Exercise 2.2

2.2 State the purpose of the `OTHERWISE` clause in a `CASE` structure.

[1]

`OTHERWISE`

Exercise 2.2

2.2 State the purpose of the OTHERWISE clause in a CASE structure.

[1]

OTHERWISE

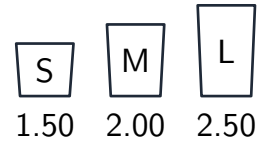
2.2

- it gives the statement(s) to run when the variable matches **none** of the listed values

Exercise 2.3

2.3 A shop sells drinks in three sizes. The **CHAR** variable **Size** holds 'S', 'M' or 'L'. A small drink costs 1.50, a medium drink 2.00 and a large drink 2.50. Complete the **CASE** structure so that it stores the correct price. [3]

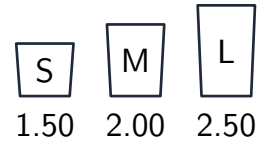
```
CASE OF Size
    'S' : Price ← .....
    ..... : Price ← 2.00
    'L' : Price ← 2.50
    ..... : OUTPUT "No such size"
ENDCASE
```



Exercise 2.3

2.3 A shop sells drinks in three sizes. The `CHAR` variable `Size` holds 'S', 'M' or 'L'. A small drink costs 1.50, a medium drink 2.00 and a large drink 2.50. Complete the `CASE` structure so that it stores the correct price. [3]

```
CASE OF Size
    'S' : Price ← .....
    ..... : Price ← 2.00
    'L' : Price ← 2.50
    ..... : OUTPUT "No such size"
ENDCASE
```



2.3

- the price of a small drink: **1.50**
- the value for a medium drink, in single quotes because it is a `CHAR`: **'M'**
- the line for any other value: **OTHERWISE**

Exercise 2.4

2.4 Rewrite this nested IF as a CASE structure.

[3]

```
IF Grade = "A" THEN
    OUTPUT "Excellent"
ELSE
    IF Grade = "B" THEN
        OUTPUT "Good"
    ELSE
        OUTPUT "Try again"
    ENDIF
ENDIF
```



Exercise 2.4

2.4 Rewrite this nested IF as a CASE structure.

[3]

```
IF Grade = "A" THEN
    OUTPUT "Excellent"
ELSE
    IF Grade = "B" THEN
        OUTPUT "Good"
    ELSE
        OUTPUT "Try again"
    ENDIF
ENDIF
```



2.4

- CASE OF ... ENDCASE with two value branches and OTHERWISE

```
CASE OF Grade
    "A": OUTPUT "Excellent"
    "B": OUTPUT "Good"
    OTHERWISE: OUTPUT "Try again"
ENDCASE
```

Exercise 2.5 (a)

2.5 State whether each IF statement could be rewritten as a CASE structure. Give a reason for each answer. [4]

(a) IF Size = 'S' THEN ... ELSE IF Size = 'M' THEN ... ELSE IF Size = 'L' THEN ...

☐ IF ☒ CASE

Exercise 2.5 (a)

2.5 State whether each IF statement could be rewritten as a CASE structure. Give a reason for each answer. [4]

(a) IF Size = 'S' THEN ... ELSE IF Size = 'M' THEN ... ELSE IF Size = 'L' THEN ...

☐ IF ☒ CASE

(a) **yes**

- every condition tests the same variable, **Size**, against one value

Exercise 2.5 (b)

2.5 State whether each IF statement could be rewritten as a CASE structure. Give a reason for each answer. [4]

(b) IF Height > 1.2 AND Age >= 10 THEN ...

Exercise 2.5 (b)

2.5 State whether each IF statement could be rewritten as a CASE structure. Give a reason for each answer. [4]

(b) IF Height > 1.2 AND Age >= 10 THEN ...

(b) **no**

- the condition tests two different variables, and a CASE structure can look at only one

Exercise 3.1

3.1 A menu uses a number **Choice**. Write a **CASE** structure: $1 \rightarrow \text{"New"}$, $2 \rightarrow \text{"Open"}$, $3 \rightarrow \text{"Save"}$, otherwise $\rightarrow \text{"Invalid"}$. [4]

1 New
2 Open
3 Save

Exercise 3.1

3.1 A menu uses a number **Choice**. Write a **CASE** structure: $1 \rightarrow \text{"New"}$, $2 \rightarrow \text{"Open"}$, $3 \rightarrow \text{"Save"}$, otherwise $\rightarrow \text{"Invalid"}$. [4]

1 New 2 Open 3 Save

3.1

- **CASE OF ... ENDCASE** with three branches, the right outputs, and **OTHERWISE**

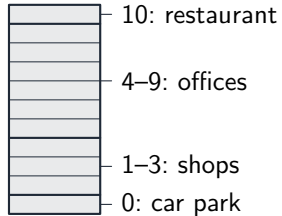
```
CASE OF Choice
  1: OUTPUT "New"
  2: OUTPUT "Open"
  3: OUTPUT "Save"
  OTHERWISE: OUTPUT "Invalid"
ENDCASE
```

Exercise 3.2

3.2 A lift (elevator) program stores the floor number in the integer variable **Floor**. The building has these floors:

- floor 0 is the car park
- floors 1 to 3 are shops
- floors 4 to 9 are offices
- floor 10 is the restaurant.

Write a **CASE** structure that outputs the use of the floor, and outputs "No such floor" for any other number. [5]

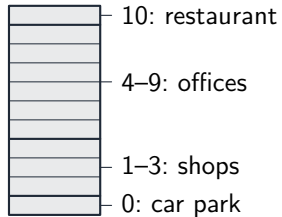


Exercise 3.2

3.2 A lift (elevator) program stores the floor number in the integer variable **Floor**. The building has these floors:

- floor 0 is the car park
- floors 1 to 3 are shops
- floors 4 to 9 are offices
- floor 10 is the restaurant.

Write a **CASE** structure that outputs the use of the floor, and outputs "No such floor" for any other number. [5]



3.2

- CASE OF Floor and ENDCASE

Full solution 3.2

3.2

- CASE OF Floor and ENDCASE
- the single values 0 and 10, each with the correct output
- the range 1 TO 3
- the range 4 TO 9
- OTHERWISE with "No such floor"

```
CASE OF Floor
  0 : OUTPUT "Car park"
  1 TO 3 : OUTPUT "Shops"
  4 TO 9 : OUTPUT "Offices"
  10 : OUTPUT "Restaurant"
  OTHERWISE : OUTPUT "No such floor"
ENDCASE
```

Exercise 3.3 (a)

3.3 Study this CASE structure.

```
CASE OF Choice
```

```
    'A' : Total  $\leftarrow$  Total + 5
```

```
    'B' : Total  $\leftarrow$  Total + 10
```

```
    OTHERWISE : Total  $\leftarrow$  0
```

```
ENDCASE
```

Choice

Total

(a) Rewrite it as nested IF statements.

[4]

Exercise 3.3 (a)

3.3 Study this CASE structure.

```
CASE OF Choice
```

```
    'A' : Total  $\leftarrow$  Total + 5
```

```
    'B' : Total  $\leftarrow$  Total + 10
```

```
    OTHERWISE : Total  $\leftarrow$  0
```

```
ENDCASE
```

Choice

Total

(a) Rewrite it as nested IF statements.

[4]

(a) IF Choice = 'A' THEN with the first assignment

- a nested IF Choice = 'B' THEN in the ELSE part, with the second assignment
- the OTHERWISE statement in the last ELSE part
- two ENDIFs

```
IF Choice = 'A' THEN
```

Full solution on next slide

Full solution 3.3 (a)

(a) IF Choice = 'A' THEN with the first assignment

- a nested IF Choice = 'B' THEN in the ELSE part, with the second assignment
- the OTHERWISE statement in the last ELSE part
- two ENDIFs

```
IF Choice = 'A' THEN
    Total ← Total + 5
ELSE
    IF Choice = 'B' THEN
        Total ← Total + 10
    ELSE
        Total ← 0
    ENDIF
ENDIF
```

Exercise 3.3 (b)

3.3 Study this CASE structure.

```
CASE OF Choice
```

```
    'A' : Total  $\leftarrow$  Total + 5
```

```
    'B' : Total  $\leftarrow$  Total + 10
```

```
    OTHERWISE : Total  $\leftarrow$  0
```

```
ENDCASE
```

Choice

Total

(b) Give **two** reasons why a programmer may prefer the CASE version.

[2]

Exercise 3.3 (b)

3.3 Study this CASE structure.

```
CASE OF Choice
  'A' : Total ← Total + 5
  'B' : Total ← Total + 10
  OTHERWISE : Total ← 0
ENDCASE
```

Choice

Total

(b) Give **two** reasons why a programmer may prefer the CASE version.

[2]

(b) any **two** of:

- it is shorter and easier to read, with no deep nesting
- it is easier to change: a new choice is one new line
- there is less chance of an error such as a missing ENDIF

Exercise 3.4 (a)

3.4 A calculator program inputs two real numbers, **First** and **Second**, and a character **Operator**, which should be '+', '-', '*' or '/'. It outputs the result of the calculation. If the operator is not one of the four, it outputs "Unknown operator".

(a) Write pseudocode for the program, using a **CASE** structure. Declare your variables.[7]



Exercise 3.4 (a)

3.4 A calculator program inputs two real numbers, **First** and **Second**, and a character **Operator**, which should be '+', '-', '*' or '/'. It outputs the result of the calculation. If the operator is not one of the four, it outputs "Unknown operator".

(a) Write pseudocode for the program, using a **CASE** structure. Declare your variables.[7]



(a) the variables declared with suitable types, including a **CHAR** for the operator

- the three inputs
- **CASE OF Operator**, closed with **ENDCASE**
- the four values in single quotes, each followed by a colon
- the correct calculation for each operator

Full solution on next slide

Full solution 3.4 (a)

(a) the variables declared with suitable types, including a **CHAR** for the operator

- the three inputs
- **CASE OF Operator**, closed with **ENDCASE**
- the four values in single quotes, each followed by a colon
- the correct calculation for each operator
- **OTHERWISE** with the message
- the result is output (only when the operator is known)

```
DECLARE First, Second : REAL
DECLARE Operator : CHAR
INPUT First
INPUT Operator
INPUT Second
CASE OF Operator
    '+' : OUTPUT First + Second
    '-' : OUTPUT First - Second
    '*' : OUTPUT First * Second
    '/' : OUTPUT First / Second
    OTHERWISE : OUTPUT "Unknown operator"
ENDCASE
```

Exercise 3.4 (b)

3.4 A calculator program inputs two real numbers, **First** and **Second**, and a character **Operator**, which should be '+', '-', '*' or '/'. It outputs the result of the calculation. If the operator is not one of the four, it outputs "Unknown operator".

(b) The program must not divide by zero. **Explain** why this check cannot be written as one more line of the **CASE** structure, and state how it should be written. [3]

Exercise 3.4 (b)

3.4 A calculator program inputs two real numbers, **First** and **Second**, and a character **Operator**, which should be '+', '-', '*' or '/'. It outputs the result of the calculation. If the operator is not one of the four, it outputs "Unknown operator".

(b) The program must not divide by zero. **Explain** why this check cannot be written as one more line of the **CASE** structure, and state how it should be written. [3]

(b) the **CASE** structure looks at the variable **Operator** only

- the check is about a **different** variable, **Second**, so it cannot be one of the values of **Operator**
- it is written as an **IF** statement inside the '/' line (a nested selection): **IF Second = 0 THEN** output an error message, **ELSE** output **First / Second**