Selection

Ref As Pseudocode

IF, THEN, ELSE, ENDIF

Selection means the program chooses what to do next, depending on a condition . A condition is an expression that is `TRUE` or `FALSE`.

■ IF without ELSE

```
IF <condition> THEN
   <statement(s)>
ENDIF
```

```
DECLARE Temperature : INTEGER
Temperature ← 31
IF Temperature > 30 THEN
   OUTPUT "Hot day"
   OUTPUT "Drink water"
ENDIF
OUTPUT "Have a good day"
```

If the condition is `TRUE`, the statements between `THEN` and `ENDIF` run. If it is `FALSE`, they are skipped. Either way, the program then carries on after `ENDIF`.

■ IF with ELSE

```
IF <condition> THEN
   <statement(s)>
ELSE
   <statement(s)>
ENDIF
```

```
DECLARE Mark : INTEGER
Mark ← 41
IF Mark >= 50 THEN
   OUTPUT "Pass"
ELSE
   OUTPUT "Fail"
ENDIF
```

Exactly **one** of the two branches runs: never both, never neither.

exercises.lol exercises.com.cn

The 9618 layout

- `THEN` goes at the **end of the `IF` line**.
- `ELSE` and `ENDIF` line up with the `IF`.
- The statements in each branch are indented.

Common mistakes

- Forgetting `ENDIF`. Every `IF` needs one.
- `IF Mark ← 50 THEN`. The arrow stores a value; a condition compares with `=`.
- Writing `THEN` on its own line below the `IF`. That is the **IGCSE 0478** layout. In 9618 it goes on the `IF` line.

Nested IF

An `IF` inside another `IF` is called a **nested** `IF`. Use it when there are more than two outcomes, or when a second question only makes sense after the first.

```
DECLARE Mark : INTEGER
Mark ← 64
IF Mark >= 70 THEN
   OUTPUT "Distinction"
ELSE
   IF Mark >= 50 THEN
      OUTPUT "Merit"
   ELSE
      OUTPUT "Fail"
   ENDIF
ENDIF
```

- The inner `IF` sits inside the outer `ELSE`, and it has its **own** `ENDIF`.
- Count them: two `IF`s need two `ENDIF`s.
- **Test the highest boundary first.** The inner test `Mark >= 50` only runs when the mark is already below 70, so it does not need `AND Mark < 70`.

There is **no** `ELSEIF` or `ELIF` keyword in Cambridge pseudocode. Write `ELSE`, then a new `IF` inside it, as above.

exercises.lol exercises.com.cn

A second question inside the first

Sometimes the inner `IF` sits in the `THEN` branch:

```
DECLARE ChallengerScore : INTEGER
DECLARE ChampionScore : INTEGER
DECLARE HighestScore : INTEGER
ChallengerScore ← 95
ChampionScore ← 80
HighestScore ← 90
IF ChallengerScore > ChampionScore THEN
   IF ChallengerScore > HighestScore THEN
      OUTPUT "New champion and new highest score"
   ELSE
      OUTPUT "New champion"
   ENDIF
ELSE
   OUTPUT "The champion stays"
ENDIF
```

Common mistakes

- Missing an inner `ENDIF`. Then every line after it seems to belong to the wrong block.
- Testing the smallest boundary first: with `IF Mark >= 50` first, a mark of 85 prints `Merit` and never reaches the `Distinction` test.
- Writing `ELSE IF` and closing it with only one `ENDIF`. Each `IF` closes separately.

CASE OF

`CASE OF` chooses one branch by the **value of one variable**. It is clearer than many nested `IF`s when you test the same variable again and again.

```
CASE OF <identifier>
   <value 1> : <statement(s)>
   <value 2> : <statement(s)>
   OTHERWISE : <statement(s)>
ENDCASE
```


exercises.lol exercises.com.cn

```
DECLARE Move : CHAR
DECLARE Position : INTEGER
Position ← 50
Move ← 'A'
CASE OF Move
   'W' : Position ← Position - 10
   'S' : Position ← Position + 10
   'A' : Position ← Position - 1
   'D' : Position ← Position + 1
   OTHERWISE : OUTPUT "Unknown key"
ENDCASE
OUTPUT Position
```

- Each line is a **value**, a colon `:`, then what to do.
- The cases are tested **in order**. The first one that matches runs, and then the whole `CASE` is finished. The rest are not tested.
- `OTHERWISE` is optional. If it is there, it must be **last**, and it runs when no value matched. In 9618 it takes a colon: `OTHERWISE :`.

■ Ranges with TO

In 9618 a case can be a **range** range : `<value1> TO <value2>`. Both ends are included.

```
DECLARE Mark : INTEGER
Mark ← 64
CASE OF Mark
   70 TO 100 : OUTPUT "A"
   60 TO 69 : OUTPUT "B"
   50 TO 59 : OUTPUT "C"
   OTHERWISE : OUTPUT "U"
ENDCASE
```

Several statements in one case

A case may run more than one statement. Put them on the following lines, indented:

```
DECLARE Choice : INTEGER
DECLARE Balance : REAL
Balance ← 100.0
Choice ← 2
CASE OF Choice
   1 : OUTPUT "Balance: ", Balance
   2 :
      Balance ← Balance - 20.0
      OUTPUT "Took out 20. Balance: ", Balance
   OTHERWISE : OUTPUT "Not a menu choice"
ENDCASE
```

IF or CASE?

- One variable compared with **fixed values or ranges**: `CASE OF` is often clearer.
- A condition with **more than one variable**, or with `AND` / `OR`: you need `IF`.

Practice

1. `Day` holds a number from 1 to 7. Using `CASE OF`, output `"Weekday"` for 1 to 5, `"Weekend"` for 6 or 7, and `"Error"` for anything else.

2. Output `"Child"` for ages 0–12, `"Teen"` for 13–19 and `"Adult"` for 20 or more, using nested `IF`.

<details> <summary>Model answer</summary>

```
DECLARE Day : INTEGER
Day ← 6
CASE OF Day
   1 TO 5 : OUTPUT "Weekday"
   6 TO 7 : OUTPUT "Weekend"
   OTHERWISE : OUTPUT "Error"
ENDCASE
```

```
DECLARE Age : INTEGER
Age ← 15
IF Age <= 12 THEN
   OUTPUT "Child"
ELSE
   IF Age <= 19 THEN
      OUTPUT "Teen"
   ELSE
      OUTPUT "Adult"
   ENDIF
ENDIF
```

</details>

Common mistakes

- Using `CASE OF` with a condition like `Mark >= 50 :`. A case is a value or a range, never a comparison.
- Putting `OTHERWISE` anywhere but last.
- Forgetting `ENDCASE`.
- `OTHERWISE` with no colon. That is the IGCSE form; 9618 writes `OTHERWISE :`.