# Input, output and operators

Ref As Pseudocode

## INPUT and OUTPUT

### INPUT

`INPUT` reads a value from the user and stores it in a variable.

```
INPUT <identifier>
```

The identifier must be a variable. It can also be one element of an array or one field of a record.

```
DECLARE Age : INTEGER
OUTPUT "How old are you?"
INPUT Age
OUTPUT "Next year you will be ", Age + 1
```

When you press Run, the page asks you to type the input first.

- Show a **prompt** first: an `OUTPUT` that tells the user what to type. Mark schemes often give a mark for it.
- `INPUT` takes **one** variable. To read two values, write two `INPUT` lines.

### OUTPUT

`OUTPUT` shows one or more values on the screen. Separate several values with **commas**.

```
OUTPUT <value(s)>
```

```
DECLARE Lives : INTEGER
DECLARE Score : INTEGER
Lives ← 3
Score ← 1250
OUTPUT Score
OUTPUT "You have ", Lives, " lives left"
```

The values are printed one after another on the same line, with nothing added between them. That is why `"You have "` ends with a space.

**Common mistakes**

- `INPUT "Enter age", Age`. `INPUT` does not show a message. Use an `OUTPUT` line first.
- `OUTPUT "Total is Total"`. The word inside the quotes is just text. Put the variable outside the quotes: `OUTPUT "Total is ", Total`.



exercises.lol exercises.com.cn

- `PRINT` instead of `OUTPUT`. `PRINT` is Python, not Cambridge pseudocode.

## Arithmetic, DIV and MOD

### The arithmetic operators

| Operator | Meaning | Example | Result |
|---|---|---|---|
| `+` | add | `7 + 2` | `9` |
| `-` | subtract | `7 - 2` | `5` |
| `*` | multiply | `7 * 2` | `14` |
| `/` | divide | `7 / 2` | `3.5` |
| `DIV` | whole-number division | `7 DIV 2` | `3` |
| `MOD` | remainder after division | `7 MOD 2` | `1` |

- `/` always gives a `REAL`, even when both numbers are integers: `10 / 2` is `5.0`.
- `*` and `/` are done before `+` and `-`, as in normal maths. Use **brackets** to make the order clear: `(A + B) / 2`.

```
DECLARE A : INTEGER
DECLARE B : INTEGER
A ← 10
B ← 4
OUTPUT A + B * 2
OUTPUT (A + B) * 2
OUTPUT A / B
```

This prints `18`, `28` and `2.5`.

### DIV and MOD

`DIV` gives the **quotient** : how many whole times one number goes into another. `MOD` gives the **remainder** : what is left over. In 9618 both are written **between** their two numbers, like `+`.

```
DECLARE Minutes : INTEGER
Minutes ← 135
OUTPUT Minutes DIV 60, " hours and ", Minutes MOD 60, " minutes"
```

135 minutes is 2 hours ($2 \times 60 = 120$) with 15 minutes left over.

Three uses you will meet again and again:

- **Even or odd**: `Number MOD 2 = 0` is `TRUE` for an even number.
- **Last digit**: `Number MOD 10` gives the last digit of a positive whole number.
- **Remove the last digit**: `Number DIV 10`. So `472 DIV 10` is `47`.

```
DECLARE Number : INTEGER
DECLARE DigitSum : INTEGER
Number ← 472
DigitSum ← 0
WHILE Number > 0
   DigitSum ← DigitSum + (Number MOD 10)
   Number ← Number DIV 10
ENDWHILE
OUTPUT DigitSum
```

This adds the digits 4 + 7 + 2 and prints `13`.

**Common mistakes**

- `DIV(7, 2)` or `MOD(7, 2)`. That is the **IGCSE** form. In 9618 write `7 DIV 2`.
- Using `/` when a whole number is needed, for example for an array index. `7 / 2` is `3.5`, which is not a valid index; use `7 DIV 2`.

## Relational and logic operators

### Relational operators

A **relational operator** compares two values. The answer is always a `BOOLEAN`: `TRUE` or `FALSE`.

| Operator | Meaning |
|---|---|
| `=` | equal to |
| `<>` | not equal to |
| `<` | less than |
| `>` | greater than |
| `<=` | less than or equal to |
| `>=` | greater than or equal to |

```
DECLARE Mark : INTEGER
Mark ← 65
OUTPUT Mark >= 50
OUTPUT Mark = 100
OUTPUT Mark <> 0
```

This prints `TRUE`, `FALSE` and `TRUE`.

### Logic operators

The **logic operators** `AND`, `OR` and `NOT` join or change Boolean values.



| Expression | It is TRUE when... |
|---|---|
| `A AND B` | **both** A and B are TRUE |
| `A OR B` | **at least one** of A and B is TRUE |
| `NOT A` | A is FALSE |

A range test needs `AND` with **two** full comparisons:

```
DECLARE Age : INTEGER
DECLARE HasTicket : BOOLEAN
Age ← 15
HasTicket ← FALSE
OUTPUT Age >= 13 AND Age <= 19
OUTPUT Age < 12 OR HasTicket
OUTPUT NOT HasTicket
```

This prints `TRUE`, `FALSE` and `TRUE`.

When `AND` and `OR` are mixed, use **brackets** so the meaning is clear: `(Day = "Sat" OR Day = "Sun") AND NOT Raining`.

### Practice

**1.** What does `29 DIV 4` give? What does `29 MOD 4` give?

**2.** Write one expression that is TRUE when `Temp` is **outside** the range 18 to 25.

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

1. `29 DIV 4` is `7` and `29 MOD 4` is `1`, because $4 \times 7 = 28$ and 1 is left over.
2. `Temp < 18 OR Temp > 25`. (Also correct: `NOT (Temp >= 18 AND Temp <= 25)`.)

```
DECLARE Temp : INTEGER
Temp ← 27
OUTPUT 29 DIV 4, " ", 29 MOD 4
OUTPUT Temp < 18 OR Temp > 25
```

</details>

**Common mistakes**

- `IF 13 <= Age <= 19`. Each comparison needs its own variable: `Age >= 13 AND Age <= 19`.
- `Age >= 13 AND <= 19`. The second half has no left side. Write `Age` again.
- `!=` or `==` for "not equal" and "equal". Those are Python and C. Pseudocode uses `<>` and `=`.



exercises.lol exercises.com.cn