

Procedures and functions

Ref Pseudocode

PROCEDURE and CALL

A **procedure** is a named block of statements. Writing one once and calling it from several places is **decomposition** — the idea the whole of paper 2 rests on.

```
PROCEDURE <identifier>
    <statements>
ENDPROCEDURE
```

A procedure is **called** by name, with the keyword **CALL**:

```
PROCEDURE DefaultLine
    OUTPUT "-----"
ENDPROCEDURE

CALL DefaultLine
OUTPUT "Report"
CALL DefaultLine
```

- The definition does nothing on its own. Only **CALL** runs it.
- Control returns to the line after the **CALL** when the procedure ends.
- A procedure **does not give a value back**. If you need one, you want a function.

Common mistakes

- Writing `DefaultLine` on its own to run it. 0478 requires **CALL**.
- Leaving out **ENDPROCEDURE**.
- Expecting the procedure's own variables to exist outside it.

FUNCTION and RETURN

A **function** is a procedure that hands a value back. It is written where that value is needed — inside an **OUTPUT**, on the right of an assignment, inside a condition.

```
FUNCTION <identifier>(<parameters>) RETURNS <data type>
    <statements>
    RETURN <value>
ENDFUNCTION
```

```
FUNCTION SumSquare(Number1 : INTEGER, Number2 : INTEGER) RETURNS INTEGER
    RETURN Number1 * Number1 + Number2 * Number2
ENDFUNCTION
```

```
OUTPUT "Sum of squares = ", SumSquare(10, 20)
```

- RETURNS <data type> on the header says what type comes back. It is not optional.
- RETURN ends the function **immediately** — nothing after it runs.
- A function is **never** called with CALL; it is used wherever its value belongs.

A function inside a condition

```
FUNCTION IsEven(Number : INTEGER) RETURNS BOOLEAN
    RETURN MOD(Number, 2) = 0
ENDFUNCTION
```

```
DECLARE Index : INTEGER
FOR Index ← 1 TO 6
    IF IsEven(Index)
        THEN
            OUTPUT Index, " is even"
        ENDIF
    NEXT Index
```

Common mistakes

- CALL SumSquare(10, 20) — that throws the answer away.
- A function with no RETURN on some path. Every route out must return a value.
- Leaving RETURNS INTEGER off the header.

Parameters

A **parameter** is a value a procedure or function is given when it is called. Each one carries its own type.

```
PROCEDURE Line(Size : INTEGER)
  DECLARE Index : INTEGER
  FOR Index ← 1 TO Size
    OUTPUT "-"
  NEXT Index
ENDPROCEDURE

CALL Line(5)
CALL Line(20)
```

Several parameters are separated by commas, and the order at the call must match the order in the definition:

```
PROCEDURE Greet(Name : STRING, Times : INTEGER)
  DECLARE Index : INTEGER
  FOR Index ← 1 TO Times
    OUTPUT "Hello ", Name
  NEXT Index
ENDPROCEDURE

CALL Greet("Mei", 2)
CALL Greet("Sam", 1)
```

△ In **0478** every parameter is passed by value : the procedure works on a copy, so changing it inside changes nothing outside. To get a value back, use a **FUNCTION** and **RETURN** it. (A-Level 9618 adds **BYREF** for the other behaviour — topic 11.)

```
FUNCTION Doubled(Number : INTEGER) RETURNS INTEGER
  RETURN Number * 2
ENDFUNCTION

DECLARE Value : INTEGER
Value ← 7
Value ← Doubled(Value)
OUTPUT Value
```

Common mistakes

- `PROCEDURE Line(Size)` with no type.
- Calling with the arguments in the wrong order — `Greet(2, "Mei")` is a type error.

- Expecting a procedure to change the variable you passed in. In 0478 it cannot.