

11.3 Structured Programming

Name: _____ Class: _____ Date: _____ **Total: 17 marks**

KEY WORDS procedure 过程 • parameters 参数 • pass by reference 传引用 • pass by value 传值
• function 函数

Objective

Build the skills to answer exam questions on **structured programming** 结构化编程 — procedures, functions and parameters.

You must be able to:

- define and **call** a **procedure** and a **function**
- use **parameters**, including **by value** and **by reference**
- use the terms argument, parameter, return value, **local/global**

1 Worked examples

Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

■ Procedure vs function

A **procedure** does an action and returns **nothing**; a **function** returns a **value** used in an expression.

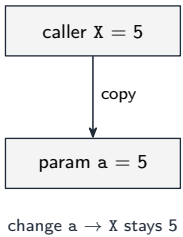
```
PROCEDURE Greet(Name : STRING)
  OUTPUT "Hello, ", Name
ENDPROCEDURE
CALL Greet("Ada")

FUNCTION Square(x : INTEGER) RETURNS INTEGER
  RETURN x * x
ENDFUNCTION
Result ← Square(5) + 1      // Result = 26
```

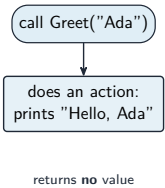
■ Parameters: by value or by reference

A **parameter** is the variable that **receives** input; the value the caller supplies is the **argument**.

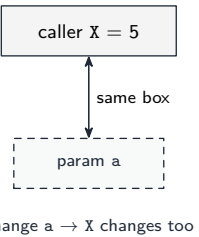
pass by value (a copy)



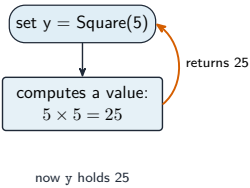
procedure



pass by reference (BYREF)



function



Procedures and functions structure a program

Swap must use **BYREF** so the new values stay in the caller:

```
PROCEDURE Swap(BYREF a : INTEGER, BYREF b : INTEGER)
  DECLARE temp : INTEGER
  temp ← a
  a ← b
  b ← temp
ENDPROCEDURE
```

■ Local, global, and when to use a subroutine

A **local** variable lives only inside its subroutine; a **global** is visible everywhere (prefer locals). Use a subroutine when logic **repeats**, has a **clear purpose**, or to **test** a part on its own.

2 Practice

Now apply the methods above.

2.1 State **one** difference between a **procedure** and a **function**. [1]

2.2 Define a procedure **Greet**(Name) that outputs "Hello, " followed by the name. [2]

2.3 Define a function **Cube**(x) that returns x cubed. [2]

2.4 State the difference between an **argument** and a **parameter**. [2]

3 Exam-style questions

3.1 Write a **function** **MaxOf**(a, b) that returns the larger of two integers. [3]

3.2 Explain **pass by value** and **pass by reference**, and state which one a **Swap** procedure must use, and why. [4]

3.3 State **two** benefits of using subroutines in a large program. [2]

3.4 State what is meant by a **local variable**. [1]

Solutions

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

2.1

- a **function returns a value** (used in an expression); a **procedure does not** return a value (it performs an action)

2.2

- PROCEDURE ... ENDPROCEDURE with a parameter

```
PROCEDURE Greet(Name : STRING)
    OUTPUT "Hello, ", Name
ENDPROCEDURE
```

2.3

- FUNCTION ... RETURNS ... ENDFUNCTION with RETURN

```
FUNCTION Cube(x : INTEGER) RETURNS INTEGER
    RETURN x * x * x
ENDFUNCTION
```

2.4

- an **argument** is the value the caller passes in
- a **parameter** is the variable inside the subroutine that **receives** it

3.1

- FUNCTION ... RETURNS INTEGER; compare; RETURN the larger

```
FUNCTION MaxOf(a : INTEGER, b : INTEGER) RETURNS INTEGER
    IF a > b THEN
        RETURN a
    ELSE
        RETURN b
    ENDIF
ENDFUNCTION
```

3.2

- **by value** — the routine gets a **copy**, so changes inside it do **not** affect the caller
- **by reference** — the routine shares the caller's actual variable, so changes **do** affect it
- Swap must use **pass by reference (BYREF)**
- so the swapped values remain in the caller's variables after the call

3.3

- any two: avoids repeating code; easier to read/maintain; can be tested on its own; supports decomposition/teamwork

3.4

- a variable declared **inside** a subroutine that exists **only while that subroutine runs** (and is not visible outside it)