2.2 PROGRAMMING TECHNIQUES · 2.2.1(C)(D)

Subroutines, parameters & scope

Modularity, functions vs procedures, parameter passing & variable scope. Spec 2.2.1(c)(d).

01 Subroutines

Function **Returns a value**: `t = add(3,5)`.

Procedure Does a **task**, may not return.

Modularity Easier to write, test, reuse, maintain.

02 Parameters & arguments

Parameter Placeholder in the **definition**: `add(a,b)`.

Argument Value passed at the **call**: `add(3,5)`.

03 Passing parameters

By value A **copy** is passed → original **unchanged**.

By reference The **original** (its location) is passed → changes **persist**.

Default Assume by value unless told.

04 Scope: local vs global

Local Inside a subroutine only; gone after.

Global Declared in main; usable everywhere.

Prefer local No interference · no clashes · modular · frees memory.

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Six slips on subroutines & scope.

01

Function returns a value; procedure performs a task.

02

Parameter = in definition; **argument** = value at the call.

03

By value = **copy** (safe); by reference = **original** (persists).

04

A **local** variable doesn't exist outside its subroutine.

05

"Compare local vs global" → benefits of locals **and** when a global is OK.

06

Prefer **parameters** over globals: self-contained, reusable, fewer bugs.