B2 PROGRAMMING · B2.3

Programming constructs

Building working programs from **sequence**, **selection** and **iteration**, then packaging code into reusable **functions**.

01 Selection

if	Run a block only if True.
if / else	One block or the other.
else if	(elif) chain conditions.
nested	An if inside another if.
case	(switch) many fixed values.

02 Iteration (loops)

while	Condition-controlled.
for	Count-controlled.
Use while	Count not known in advance.
Use for	Count known up front.
Infinite loop	Stop condition never met.

03 The three constructs

1

Sequence

Instructions run in order, top to bottom. Wrong order causes bugs.

2

Selection

A condition chooses which block runs (if / else / case).

3

Iteration

A block repeats: while (condition) or for (a set count).

04 Functions & modularization

Function	Named, reusable block of code.
Parameter	Named input in the definition.
Argument	Real value passed in a call.
return	Sends a value back.
Local / global	Inside only / whole program.

05 Worked example: count passes

count = 0	Set up counter (sequence).
for mark	Visit each mark (iteration).
if mark >= 50	Test for a pass (selection).
count = count + 1	Add one when it passes.
return count	40, 55, 60, 30 gives 2.

06 Know the difference

while vs for	Condition-controlled (repeats while true) versus count-controlled (a set number of times).	LOOPS
Parameter vs argument	The named placeholder in the definition versus the real value passed in the call.	FUNCTIONS
Local vs global	A variable that exists only inside a function versus one accessible throughout the program.	SCOPE
Selection vs iteration	Choosing a path once versus repeating a block many times.	CONSTRUCTS

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Eight things that lose marks if you slip on them. Skim before you walk in.

01

The three constructs: **sequence, selection, iteration**. Every program uses them.

02

while = condition-controlled; **for** = count-controlled. Pick the right one.

03

An **infinite loop** happens when nothing changes the condition. Always move towards stopping.

04

Parameter = placeholder in the definition; **argument** = real value in the call.

05

If a function must give a result, it needs a **return**; otherwise it hands back nothing.

06

Local variables vanish when the function ends; return a value to use it outside.

07

Use **else if** (elif) to chain conditions; use **case/switch** for one variable, many values.

08

Modularize: small named functions make code easier to read, test, reuse and update.