**🧠 Stepwise Refinement Approach**

**Definition:**  
Stepwise refinement is a *top-down design technique* where the overall task is broken down step by step into more detailed parts until each part is simple enough to be implemented directly.

**💵 (1) Check Writer Program**

**Level 0: High-Level Abstraction**

Procedure WriteCheck

Input numeric dollar amount

Convert numeric amount to words

Print the amount in check format

End Procedure

🟢 *Focus:* The main goal — take a number and print it in words.

**Level 1: Intermediate Abstraction**

Procedure WriteCheck

Step 1: GetAmountFromUser()

Step 2: SeparateAmount(dollars, cents)

Step 3: ConvertToWords(dollars, cents)

Step 4: FormatCheckOutput(words, numericAmount)

Step 5: DisplayCheck(formattedOutput)

End Procedure

🟢 *Focus:* Breaks the overall task into functional modules.

**Level 2: Detailed Abstraction**

Procedure ConvertToWords(dollars, cents)

Split dollars into thousands, hundreds, tens, and ones

For each group:

Convert numeric part into words

Add scale (e.g., “Thousand”, “Million”)

Combine dollar part with cent part

Return “One Thousand Two Hundred Fifty and 50/100 Dollars Only”

End Procedure

🟢 *Focus:* The low-level logic for one of the key functions.

**🧮 (2) Solve Roots of a Transcendental Equation (Iteratively)**

*(Example equation: f(x) = e^x - 3x = 0)*

**Level 0: High-Level Abstraction**

Procedure SolveRoot

Input function f(x)

Apply iterative method to approximate the root

Output the result

End Procedure

🟢 *Focus:* Describes the general objective — find the root numerically.

**Level 1: Intermediate Abstraction**

Procedure SolveRoot

Step 1: Read function f(x) and initial guess x0

Step 2: Choose an iteration method (Newton-Raphson or Bisection)

Step 3: Repeat until convergence:

a. Compute next approximation

b. Check error tolerance

Step 4: Output final root

End Procedure

🟢 *Focus:* Major components of numerical iteration.

**Level 2: Detailed Abstraction**

Procedure NewtonRaphson(f, f’, x0, ε)

x = x0

Repeat

x\_new = x - f(x)/f’(x)

If |x\_new - x| < ε then break

x = x\_new

Until convergence

Return x\_new

End Procedure

🟢 *Focus:* Step-by-step computation for the Newton-Raphson method.

**⚙️ (3) Simple Task Scheduling Algorithm (for OS)**

**Level 0: High-Level Abstraction**

Procedure TaskScheduler

Input list of processes

Apply scheduling policy

Output execution order

End Procedure

🟢 *Focus:* Defines the overall scheduling goal.

**Level 1: Intermediate Abstraction**

Procedure TaskScheduler

Step 1: Read process details (PID, arrival time, burst time)

Step 2: Select scheduling algorithm (e.g., FCFS, SJF, Round Robin)

Step 3: Sort or queue processes based on the chosen criteria

Step 4: Allocate CPU according to schedule

Step 5: Display process order and waiting times

End Procedure

🟢 *Focus:* Breaks the system into scheduling and result-handling modules.

**Level 2: Detailed Abstraction**

*(Example using FCFS Scheduling)*

Procedure FCFS\_Scheduler(processList)

Sort processList by arrivalTime

currentTime = 0

For each process in processList:

If currentTime < process.arrivalTime:

currentTime = process.arrivalTime

process.startTime = currentTime

process.finishTime = currentTime + process.burstTime

currentTime = process.finishTime

Calculate waiting and turnaround times

Display schedule table

End Procedure

🟢 *Focus:* Step-by-step logic for one scheduling method.