Python&Math Initiative Project

(PyMath Project)

WP2 – ALGORITHMS AND MATHEMATICS

Session: Day 2, Session 3: Introduction to Loop Statements

Presenter: Prof. Dr. Turgay Tugay BİLGİN | Bursa Technical University

Date: 17.11.2025

Contents

CONTENTS

Contents

CONTENTS

01

Introduction

What We Will Learn Today

- Review: The Limitation of 'If'
- Introduction to Loops
- The 3 Types of Loops
- The 'While' Loop (Pre-Test)
- The 'Do-While' Loop (Post-Test)
- The 'For' Loop (Counter-Controlled)
- Simple Example: Print 1 to N
- LTT Project Example
- Wrap-up & Next Steps

02

Review and Motivation

What We Learned Last Week (Recap)

Algorithms
Step-by-step instructions.

Flowgorithm Basics
Symbols for Declare, Input, Process, and Output.

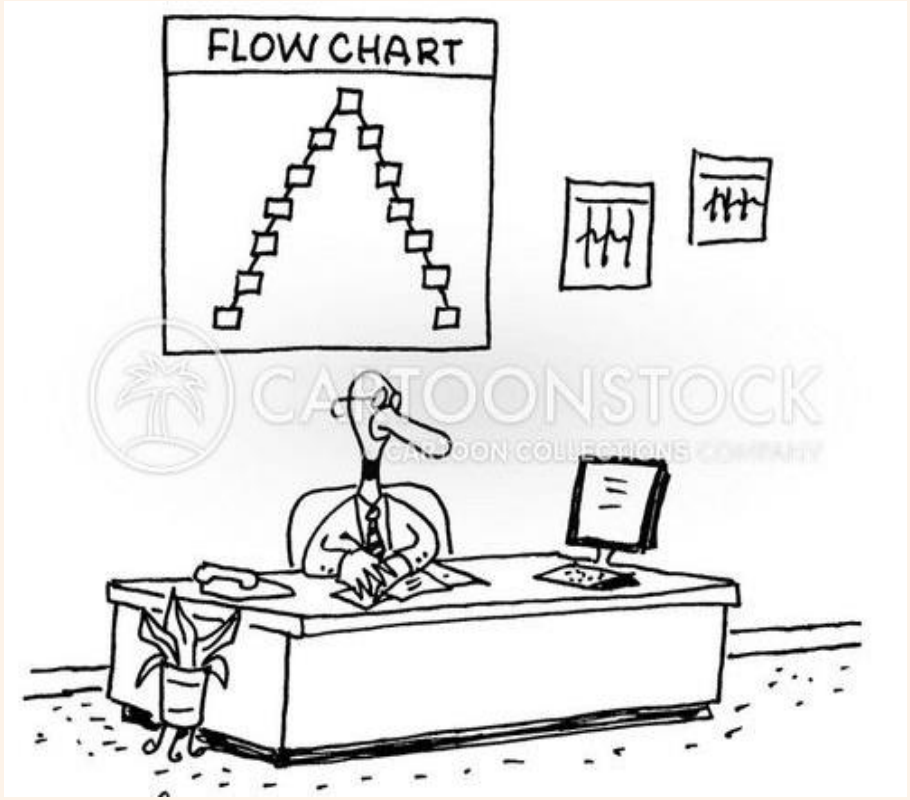
Operators
Assignment (=), Comparison (==, >, <), Logical (AND, OR).

Decision (If)
Mastered 'If' for True/False logic paths.

Our 'If/Else' Logic

Last week, we used the 'If' symbol for problems like the Even/Odd check and 'Nested If' for Piecewise Functions.

But it has a major limitation:
It does not repeat.



The Limitation of 'If'

What if we want to print numbers from 1 to 5? Or even 1 to 1,000?

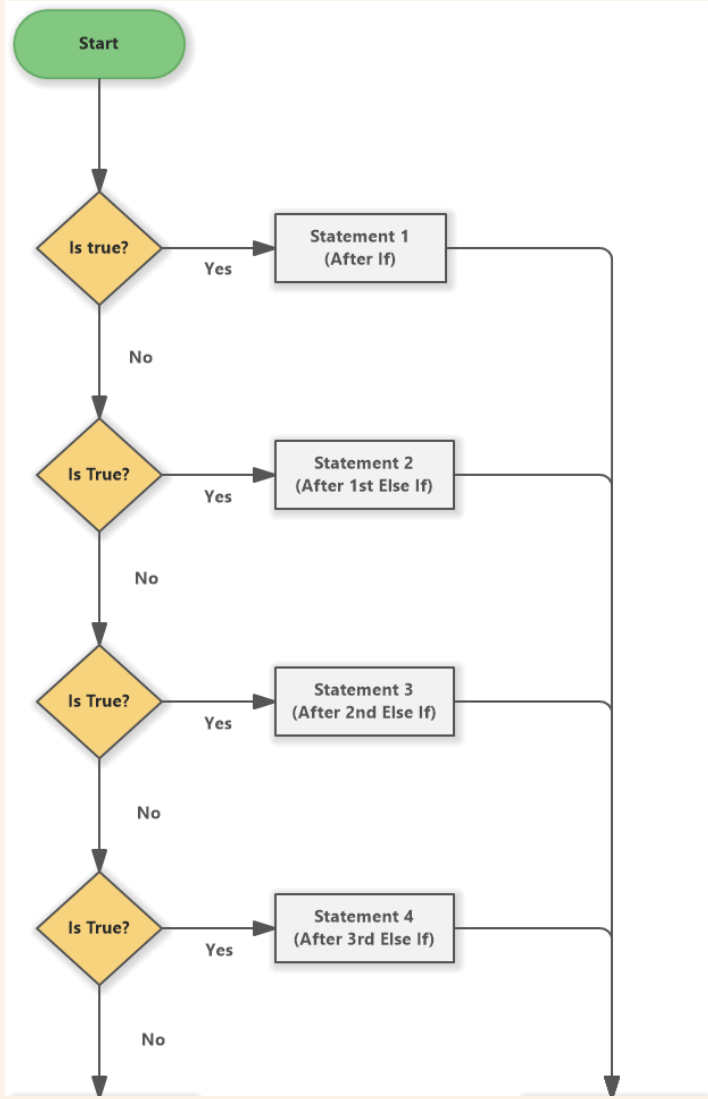
Using 'If'

Output 1
Output 2
Output 3
Output 4
Output 5

...and so on.

We cannot draw one thousand 'Output' blocks!

We need a way to repeat an action.



Without Loops

Redundant, Time-consuming, Error-prone

Write 'Output' 1000 times.

The Solution: Loops

A control structure to repeat a block of actions.

Use one 'Output' symbol inside a loop that runs 1000 times.

Efficient, Powerful, Simple

03

Session Objectives

Our Goals for This Session

Explain why
we need loops.

Understand
3 loop types.

Design
a loop flowchart.

Implement
LTT Project Example.

04

Loop Types Overview

The 3 Loop Statements in Flowgorithm

1. The 'While' Loop

Pre-Test Loop
Checks condition *before* running.

2. The 'Do-While' Loop

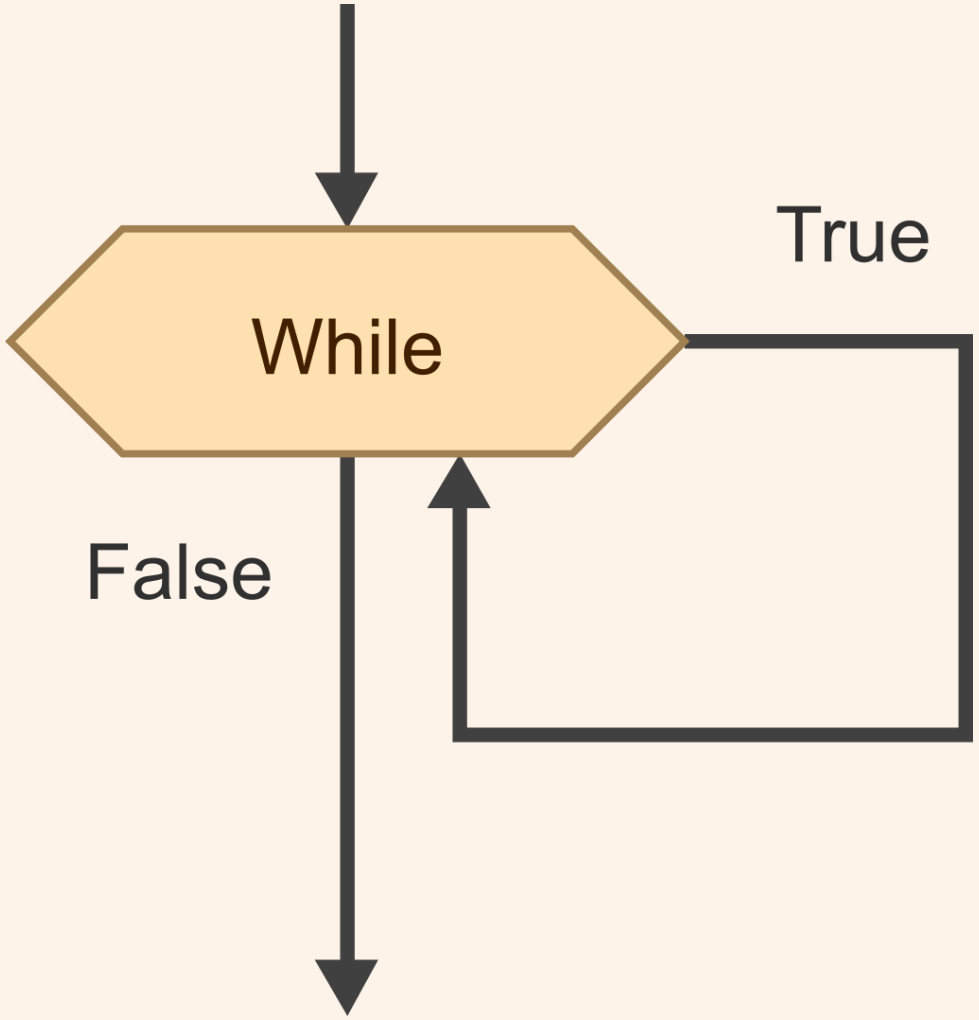
Post-Test Loop
Checks condition *after* running.

3. The 'For' Loop

Counter-Controlled
Designed for counting.

05

While Loop



The 'While' Loop

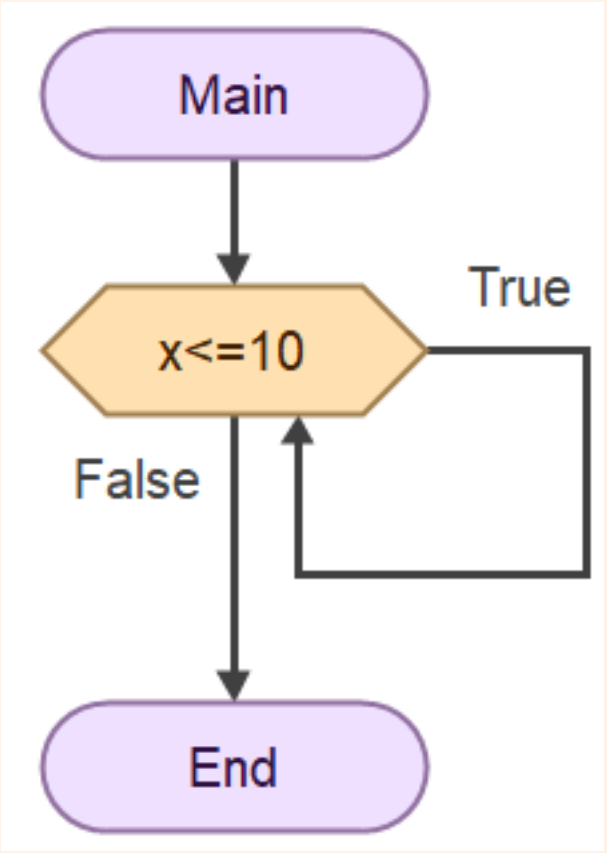
(Pre-Test Loop)

It checks a condition BEFORE it runs the code inside.

1. Check the condition (e.g., 'counter <= 5').
2. If True, execute the loop body.
3. Go back to Step 1.
4. If False, skip the loop.

Anatomy of a 'While' Loop

- Always has a **Condition** (e.g., 'x <= 10').
- One main path: the **'True' path** for the Loop Body.
- After the body, flow **must** go back to check the condition.
- When 'False', flow exits the loop.



The 'Pre-Test' Characteristic

Because it checks the condition **first**, the loop body might never run at all.

Example:

We set	Condition:	Check:	Result:
counter = 10	counter < 5	Is 10 < 5?	False

The loop is skipped. This is useful when action is conditional.

Controlling a 'While' Loop

To avoid an infinite loop, we need 3 steps with a 'counter' variable:

1. Initialize

Give the counter a starting value **before** the loop.

2. Condition

Check the counter in the 'While' symbol.

3. Increment

Change the counter's value **inside** the loop.

Example: Counting 1 to 3

Initialize: `counter = 1`

Loop: `While (counter <= 3)`

- Output: counter
- Process: `counter = counter + 1`

▶ Is 1 <= 3? True. Print 1. Counter=2.

▶ Is 2 <= 3? True. Print 2. Counter=3.

▶ Is 3 <= 3? True. Print 3. Counter=4.

■ Is 4 <= 3? False. Exit.

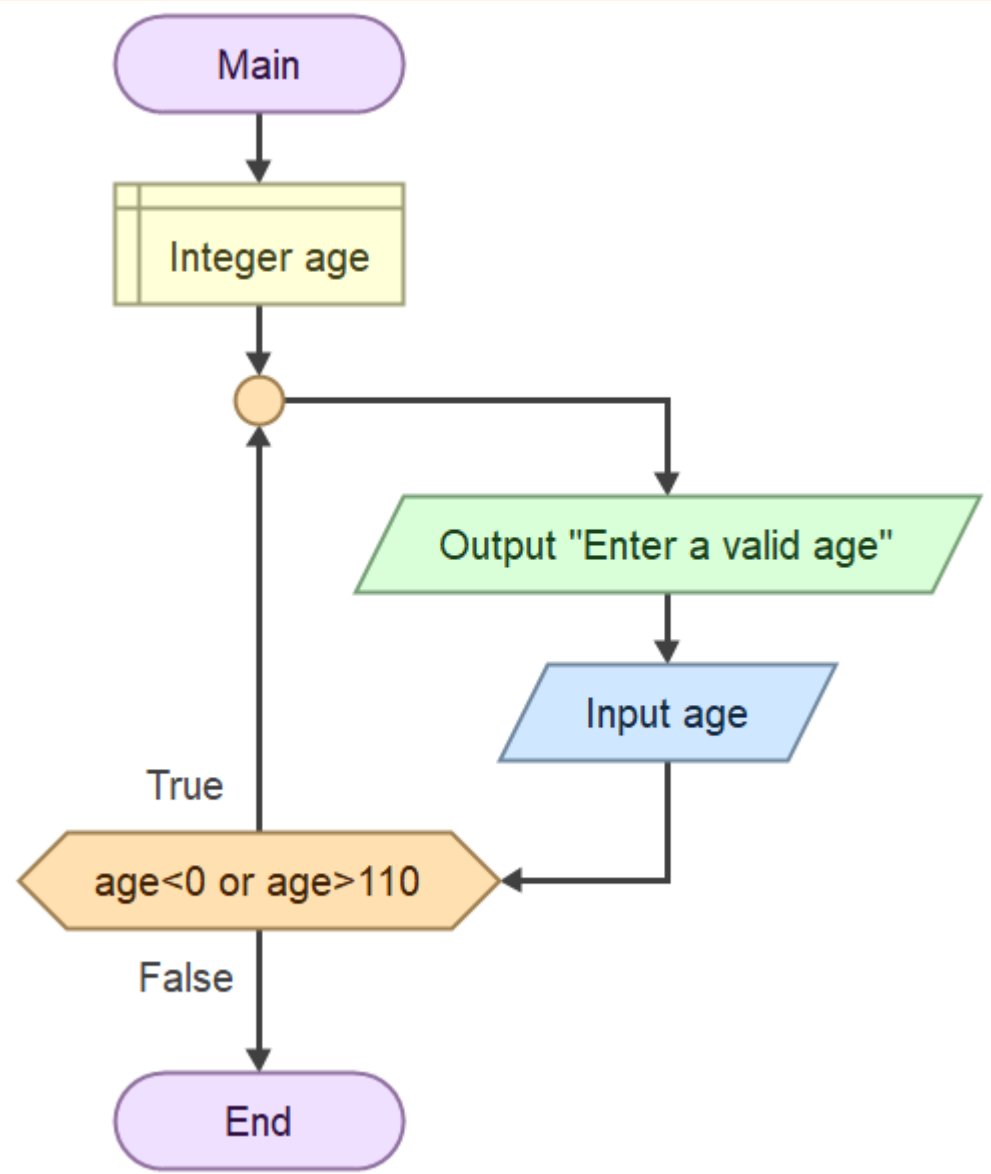
06 Do-While Loop

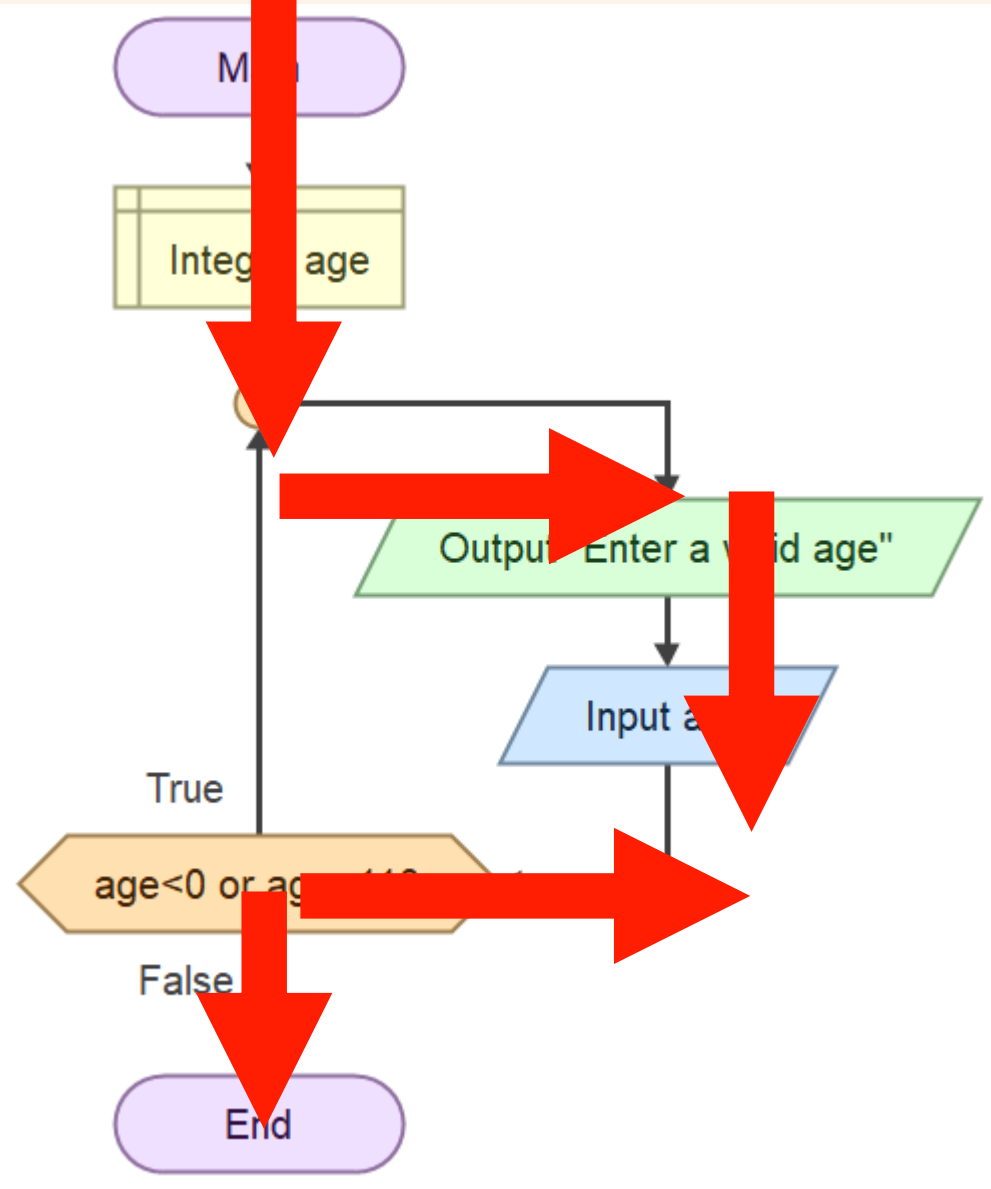
The 'Do-While' Loop

(Post-Test Loop)

It runs the code **first**, and checks the condition AFTER.

1. Execute the loop body.
2. Check the condition (e.g., 'age<0 or age>110').
3. If True, repeat the body.
4. If False, exit the loop.





The 'Post-Test' Characteristic

The most important feature:

The loop body is guaranteed to run at least one time.

Use this when you must perform an action (like asking for input) at least once.

Example: Ask for Input

Problem: Ask "Do you want to continue? (yes/no)" and repeat as long as the answer is "yes".

Loop Body (DO)
Output: "Do you want to continue?"
Input: 'answer'

→

Condition (WHILE)
'answer == "yes"'

This loop runs until the user types anything other than "yes".

'While' vs. 'Do-While'

While Loop (Pre-Test)

- ✓ Checks condition *before* running.
- ✓ Might run **zero** or more times.
- ✓ Use when not sure if loop should run.

Do-While Loop (Post-Test)

- ✓ Checks condition *after* running.
- ✓ Guaranteed to run **one** or more times.
- ✓ Use when action must run at least once.

07

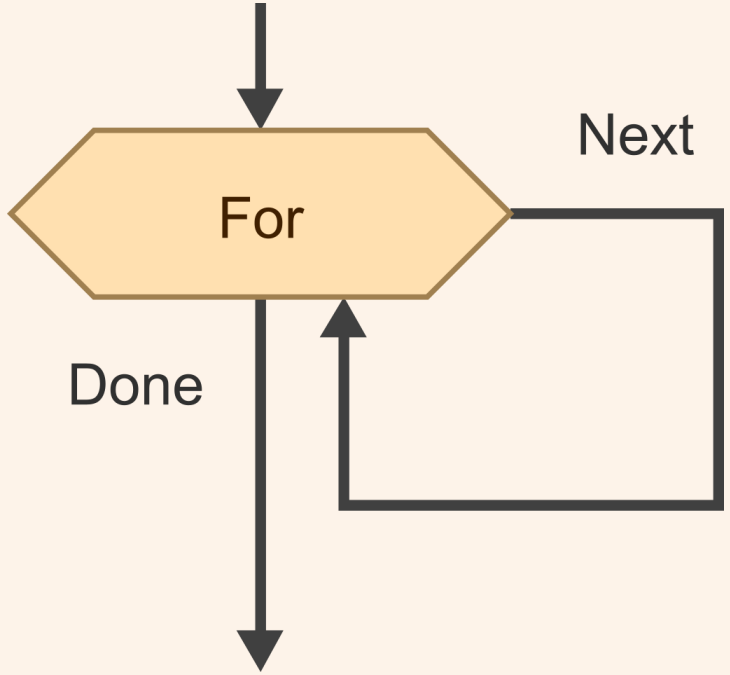
For Loop

The 'For' Loop

(Counter-Controlled Loop)

Designed specifically for **counting**.

- Combines Initialize, Condition, and Increment into **one symbol**.
- Most convenient loop for mathematical problems.



How the 'For' Loop Works

Double-click the 'For' loop symbol to set its parameters:

Variable
The counter name (e.g., 'i').

Start Value
The number to start from (e.g., 1).

End Value
The number to stop at (e.g., 'N').

Direction
'Increasing' or 'Decreasing'.

Step by
The increment value (e.g., 1 for 1,2,3... or 2 for 2,4,6...).

Example: Counting 1 to 5

Instead of 3 separate symbols, we use **one** 'For' loop.

Flowgorithm Settings:

Variable: i

Start Value: 1

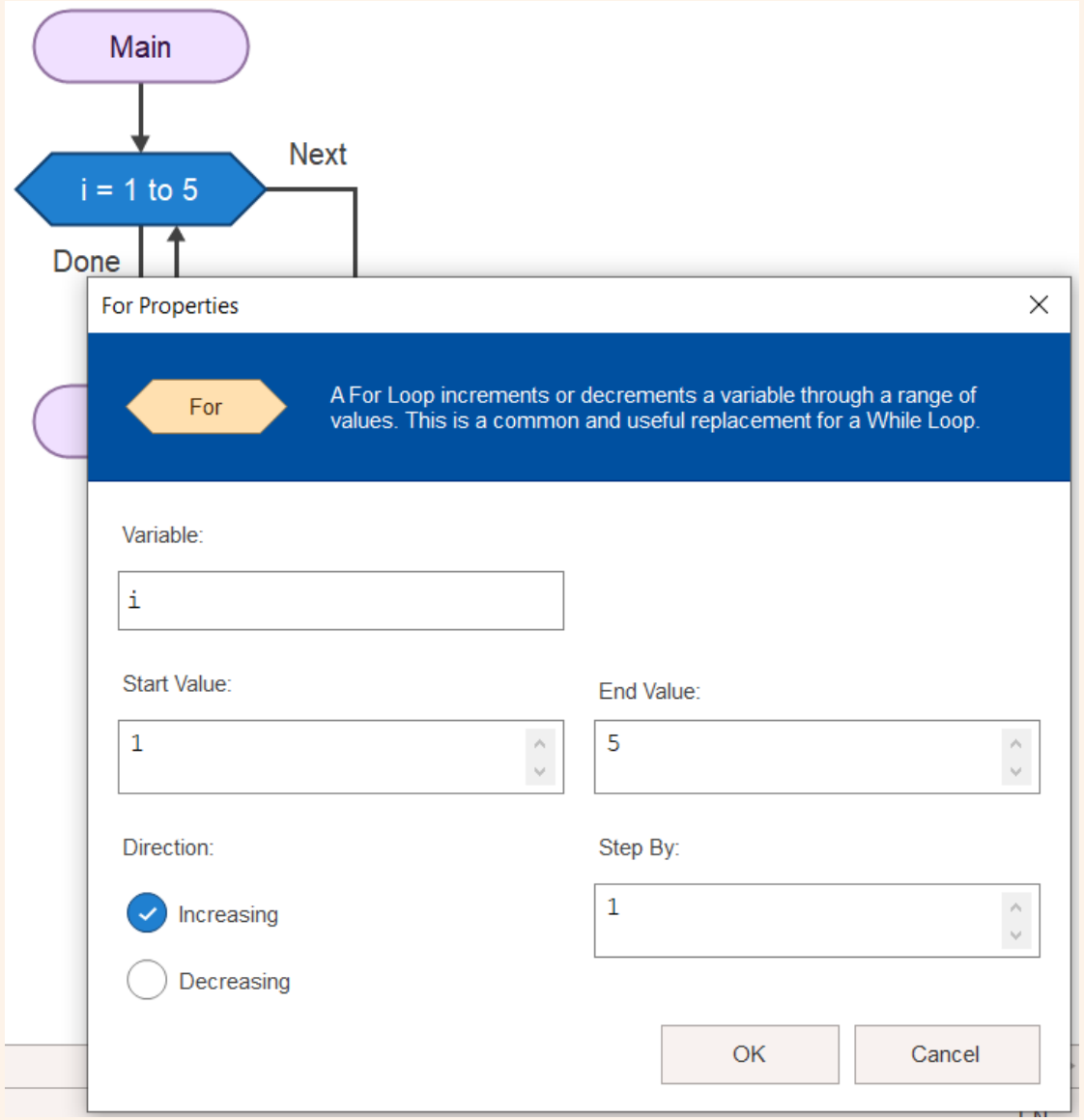
End Value: 5

Direction: Increasing

Step by: 1

Inside the loop: Output: i

This will automatically print 1, 2, 3, 4, 5 and then stop.



08

Warm-up Example

Simple Example (Warm-up)

Goal:

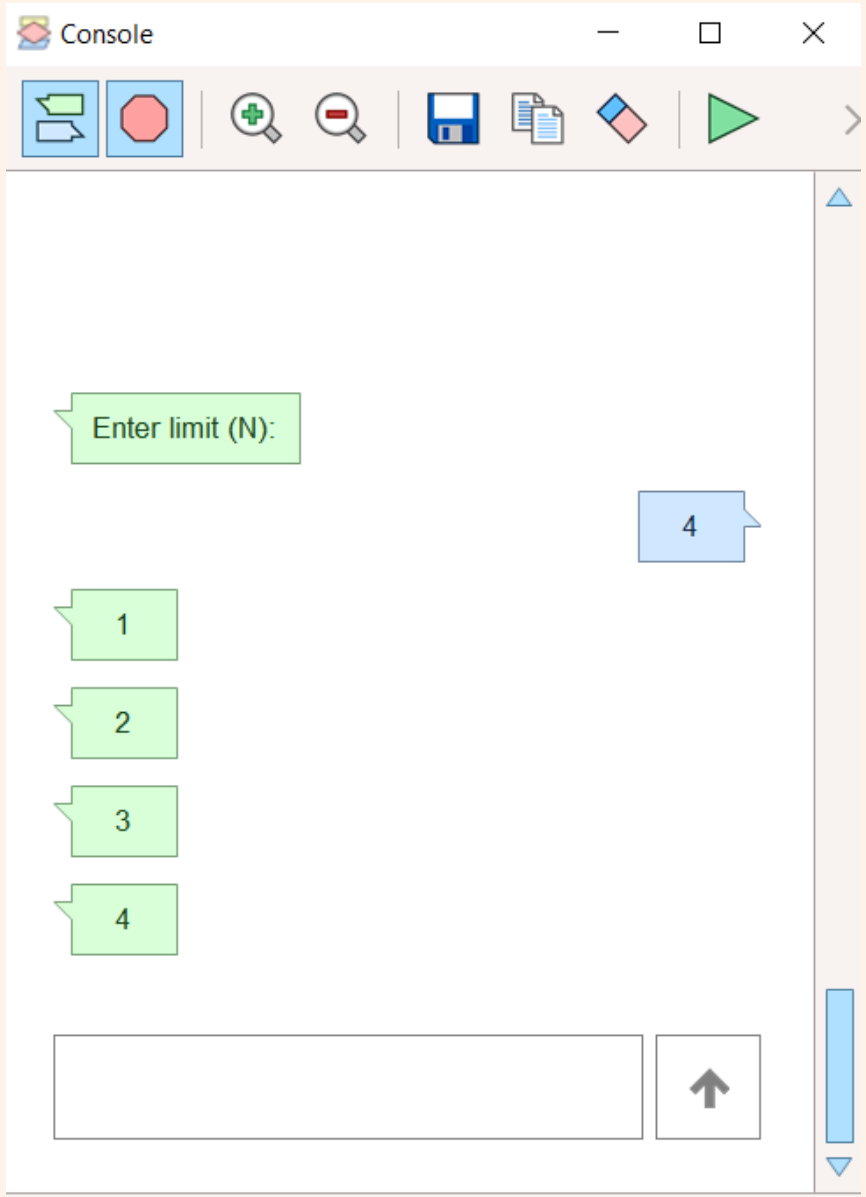
Create an algorithm using a 'For' loop.

Problem:

Ask the user for a number (N). Print all numbers from 1 up to N.

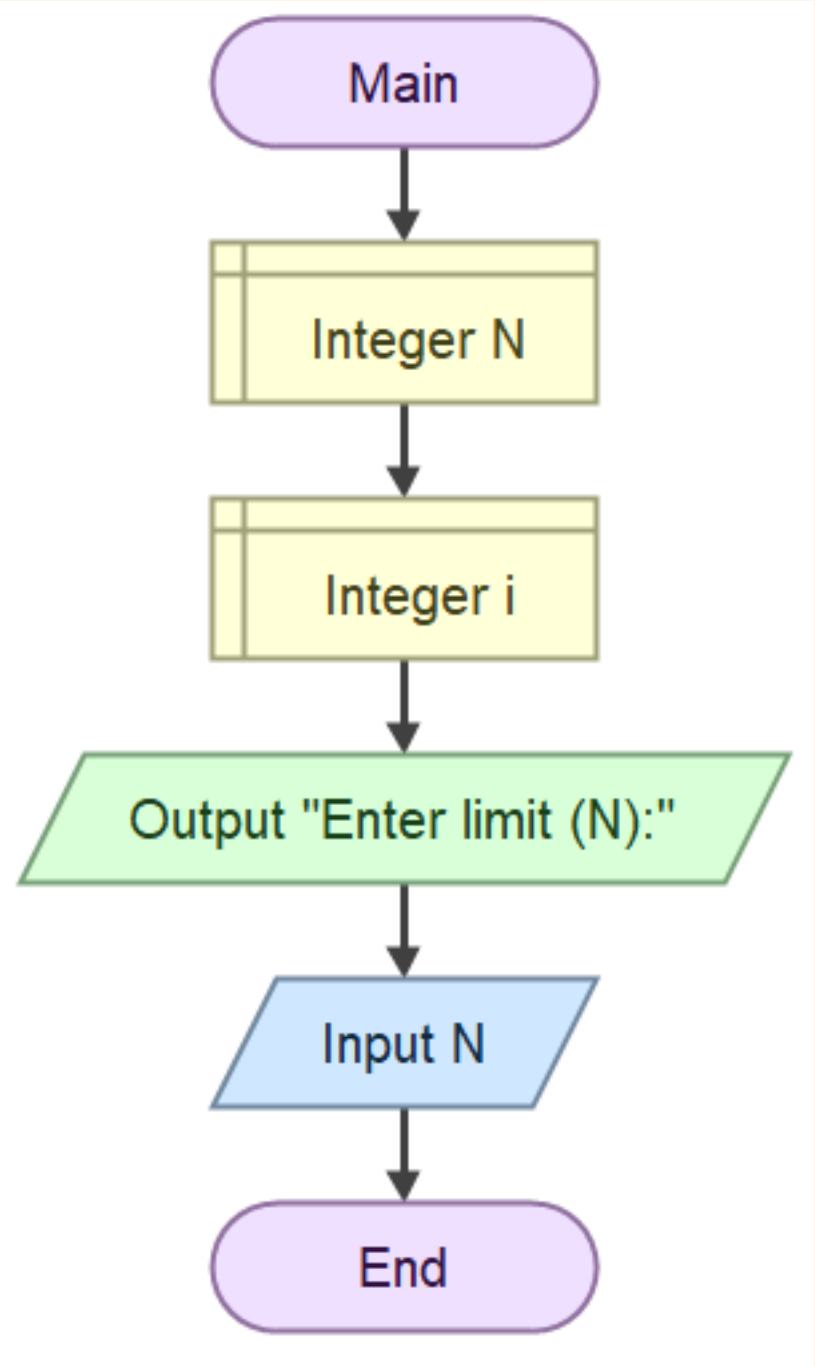
Example Run:

If User Input (N) is **4**, Output is: 1, 2, 3, 4.



Textual Algorithm: Print 1 to N

1. **Start.**
2. **Declare** Integer 'N' (end value), 'i' (counter).
3. Output: "Please enter the end number (N):"
4. Input: 'N'.
5. Start FOR loop (counter 'i' = 1 to 'N', Step 1).
6. Output: 'i'.
7. End FOR loop.
8. **Stop.**



Flowgorithm: Step 1 & 2

Start by declaring variables and getting user input.

MAIN

Declare: Integer N

Declare: Integer i

Output: "Enter limit (N):"

Input: N

Flowgorithm: Step 3

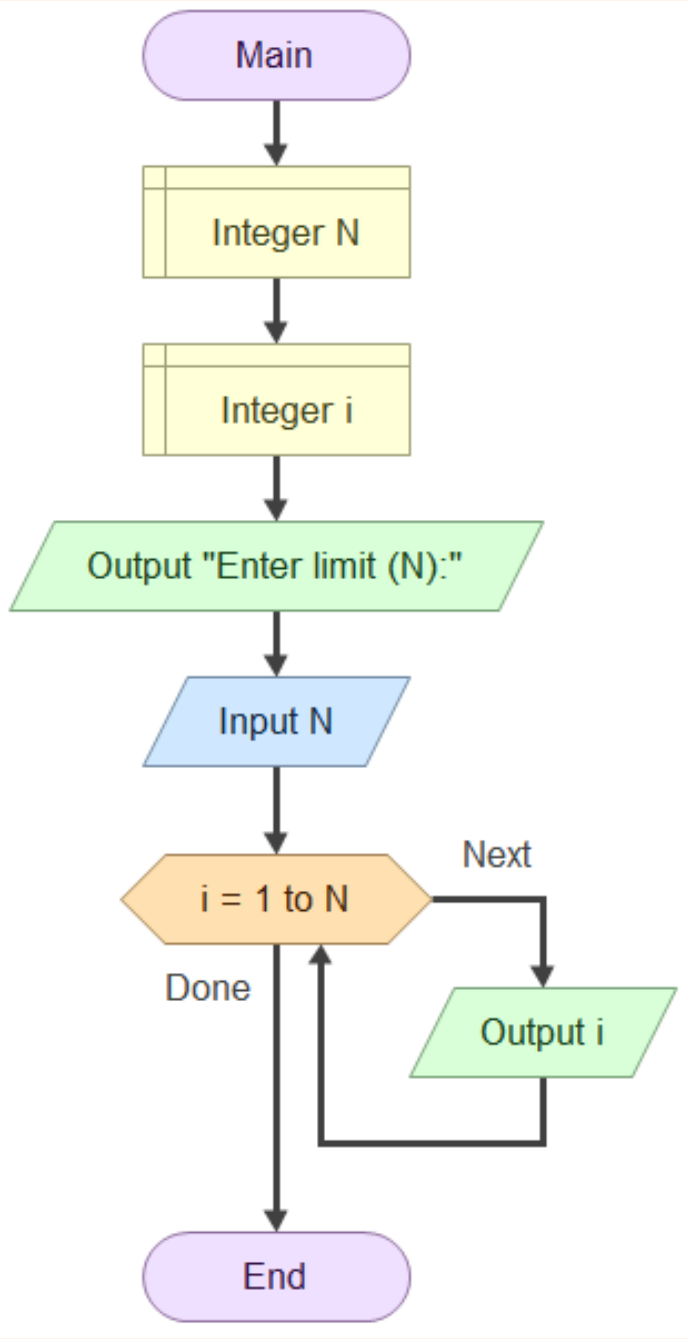
Add the 'For' loop symbol and place the 'Output' inside it.

(Flow from Input N...)

FOR i = 1 to N, Step 1

    Output: i

(Loop automatically repeats)



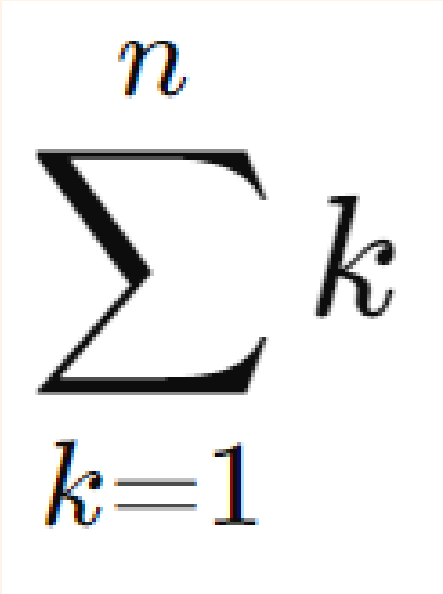
Flowgorithm: Test Run (Trace)

Let's trace the algorithm with N = 3.

Loop 1	i = 1	1 <= 3	True	Output: 1
Loop 2	i = 2	2 <= 3	True	Output: 2
Loop 3	i = 3	3 <= 3	True	Output: 3
Loop 4	i = 4	4 <= 3	False	Loop stops.

09

LTT Project Example



LTT Project Example

Goal:

"Algorithm for finding the sum of arithmetic increasing numbers"

Our Task:

Ask for a number (N). Calculate the sum of all numbers from 1 to N.

Example Run:

If N = **4**, then 1 + 2 + 3 + 4 = 10.

The Logic: From Printing to Summing

Warm-up: Printing

We printed the value of 'i'.

1, 2, 3, 4...

LTT Example: Summing

We need to add them together.

1 + 2 + 3 + 4 = 10

To do this, we need a new variable to store the running total: an Accumulator.

The 'Accumulator' Pattern

A variable (e.g., 'total') to "collect" the sum.

Step 1: Initialize

Must set 'total' to 0 **before** the loop.

Process: total = 0

Step 2: Accumulate

Add 'i' to 'total' **inside** the loop.

Process: total = total + i

Trace of the Accumulator (N = 4)

Before loop: total = 0

Loop 1 (i=1): total = 0 + 1 = 1

Loop 2 (i=2): total = 1 + 2 = 3

Loop 3 (i=3): total = 3 + 3 = 6

Loop 4 (i=4): total = 6 + 4 = 10

Final Result: The total sum is 10.

Textual Algorithm: Sum 1 to N

1. **Start.**
2. **Declare** Integer 'N', 'i'.
3. **Declare** Integer 'total'.
4. **Process:** 'total = 0'.
5. Output: "Enter limit (N):"
6. Input: 'N'.
7. Start FOR loop ('i' = 1 to 'N').
8. **Process:** 'total = total + i'.
9. End FOR loop.
10. Output: "The total sum is: " & 'total'.
11. **Stop.**

Flowgorithm: Steps 1 & 2

Declare variables and initialize the accumulator.

MAIN

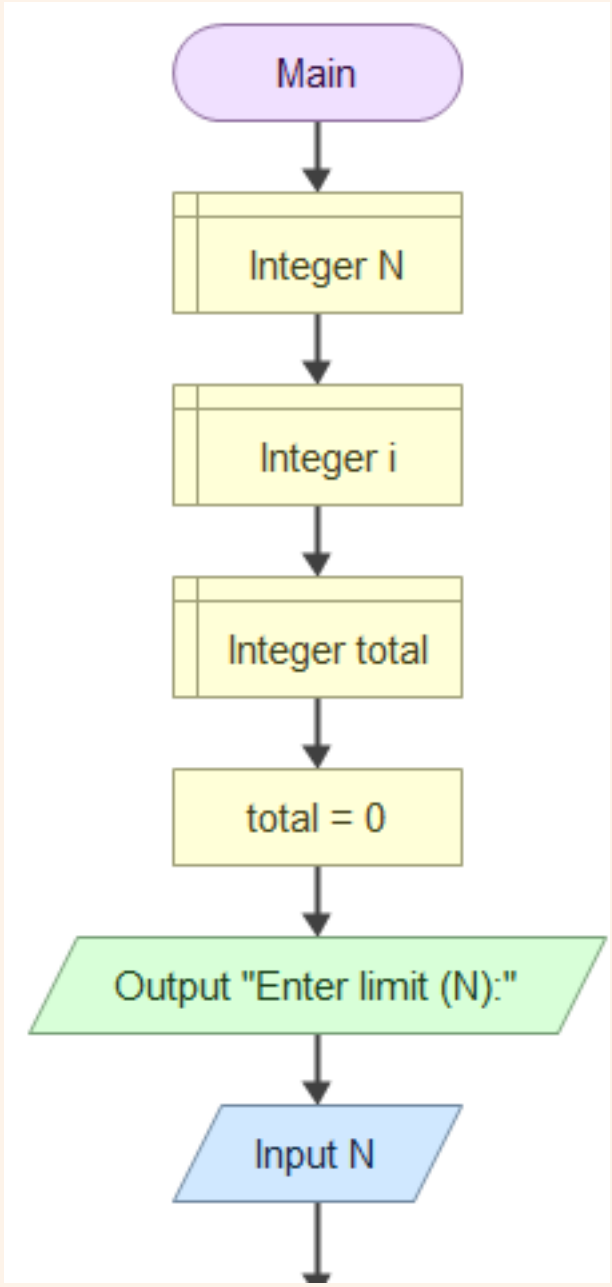
Declare: N, i, total

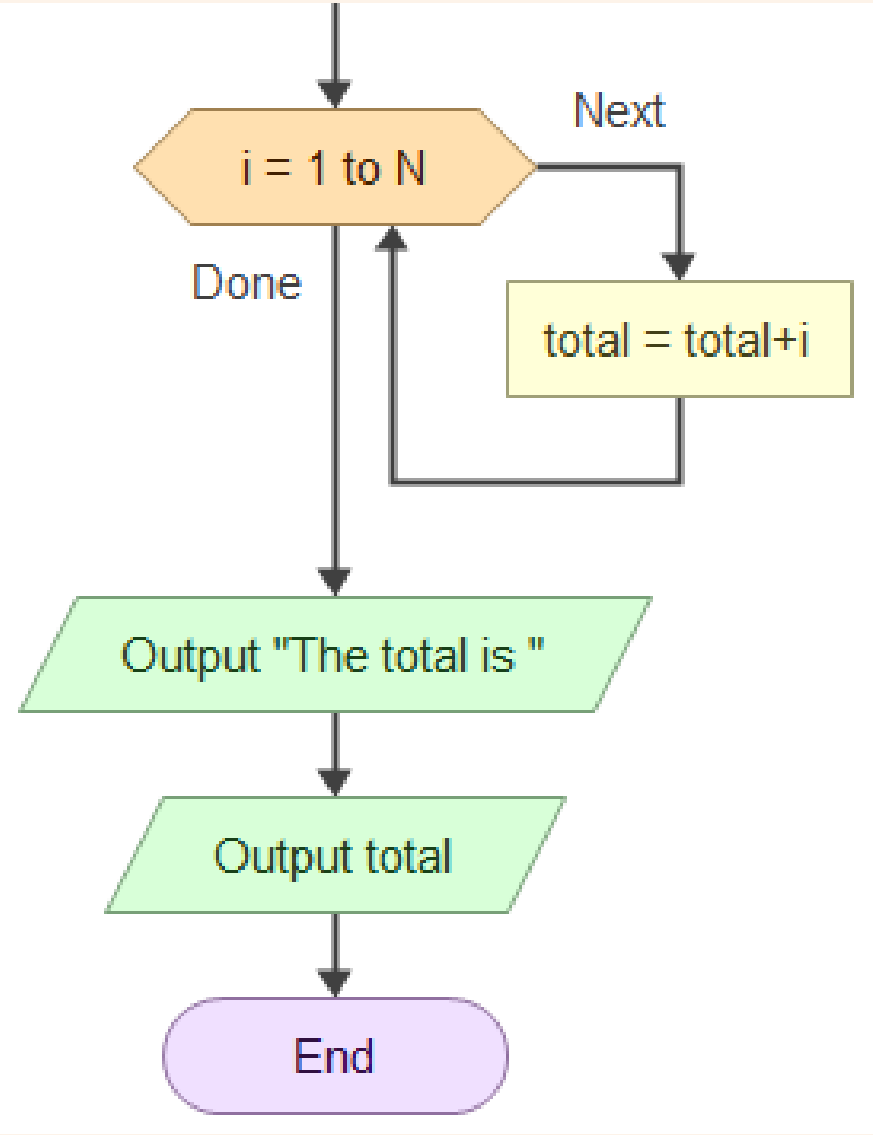
Process: total = 0

Output: "Enter limit (N):"

Input: N

We MUST set 'total' to 0 before adding!





Flowgorithm: Steps 3 & 4

Add the loop and the final output.

```
(Flow from Input N...)
FOR i = 1 to N
    Process: total = total + i
Output: "The total sum is: " & total
END
```

Completed Flowchart: Sum 1 to N

MAIN

Declare: Integer N, i, total

Process: total = 0

Output: "Enter limit (N):"

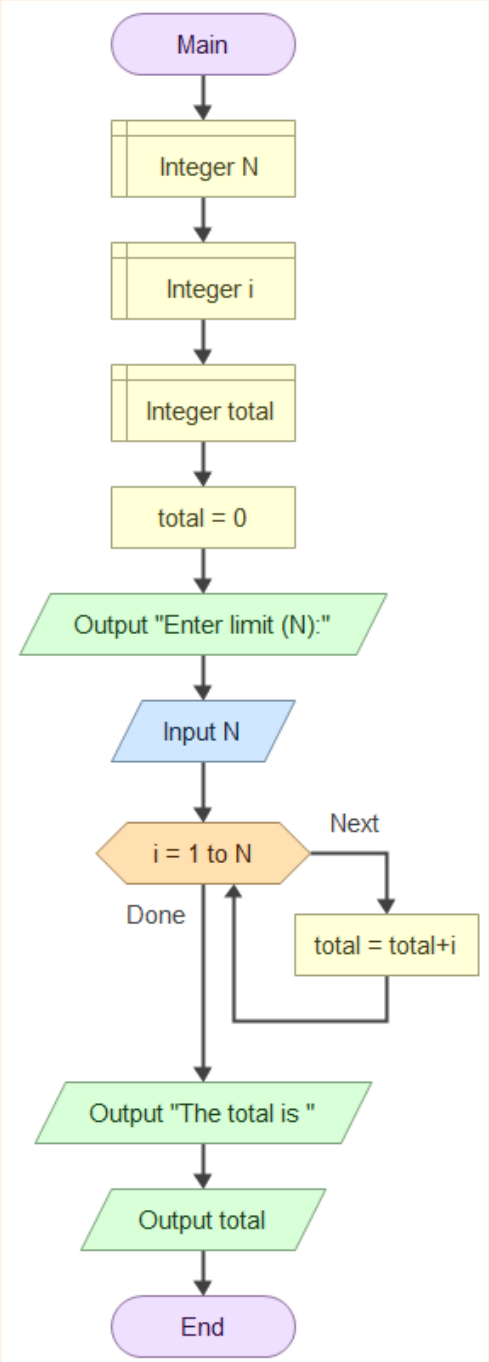
Input: N

FOR i = 1 to N, Step 1

    Process: total = total + i

Output: "The total sum is: " & total

END



Arithmetic Series Example

Goal:

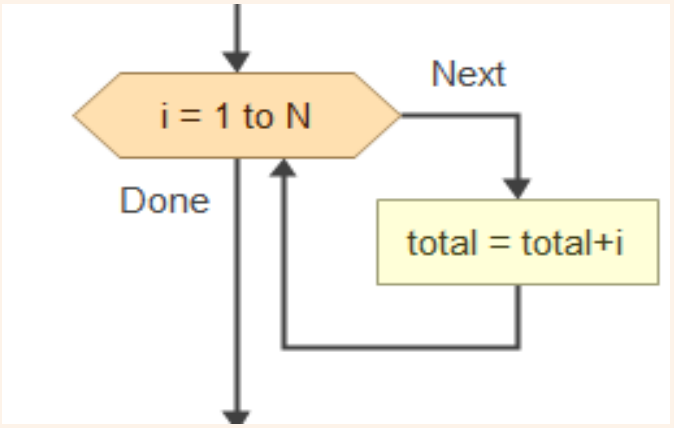
Sum of even numbers (arithmetic sequence increasing by 2).

Solution:

Change only the 'For' loop parameters. The rest of the algorithm stays the same!

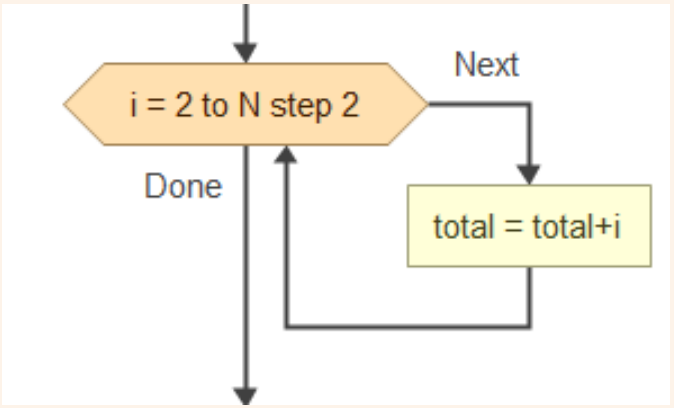
Old Loop
(Sum 1 to N)

FOR i = 1 to N, Step 1



New Loop
(Sum Even Numbers)

Start Value: 2
End Value: N
Step: 2



What We Learned

Loops repeat actions and avoid redundancy.

'While' Loop: Pre-Test, might run zero times.

'Do-While' Loop: Post-Test, runs at least once.

'For' Loop: Counter-Controlled, best for counting.

Accumulator Pattern: Variable (init to 0) to sum values inside a loop.

We successfully solved the LTT example!

10 Wrap-up

What to Expect in Session 4

Nested Loops
A loop inside another loop.

Arrays
Store a list of numbers in one variable.

LTT Project Example
Combine loops, arrays, and 'If' to find the intersection of sets.

Practical Skill
Export flowcharts as images (PNG, PDF).

THANK YOU