Python & Math Project

(PyMath Project)

WP2 - ALGORITHMS AND MATHEMATICS

Session: Day 1, Session 1: Introduction to Algorithms

Presenter: Prof.Dr Turgay Tugay BİLGİN

Bursa Technical University

Date: 10.11.2025

CONTENTS

Contents

CONTENTS

01

Kick-off & Goals

Our Project's Goal (PyMath Project)

To transform abstract learning in math classes into concrete applications.

To increase interest in mathematics by using technology and coding skills.

To develop students' problem-solving, critical thinking, and collaborative skills.

This training module (WP2) is the first step: translating mathematical thinking into algorithmic thinking.

Session 1: Objectives

By the end of this 50-minute session, you will be able to:

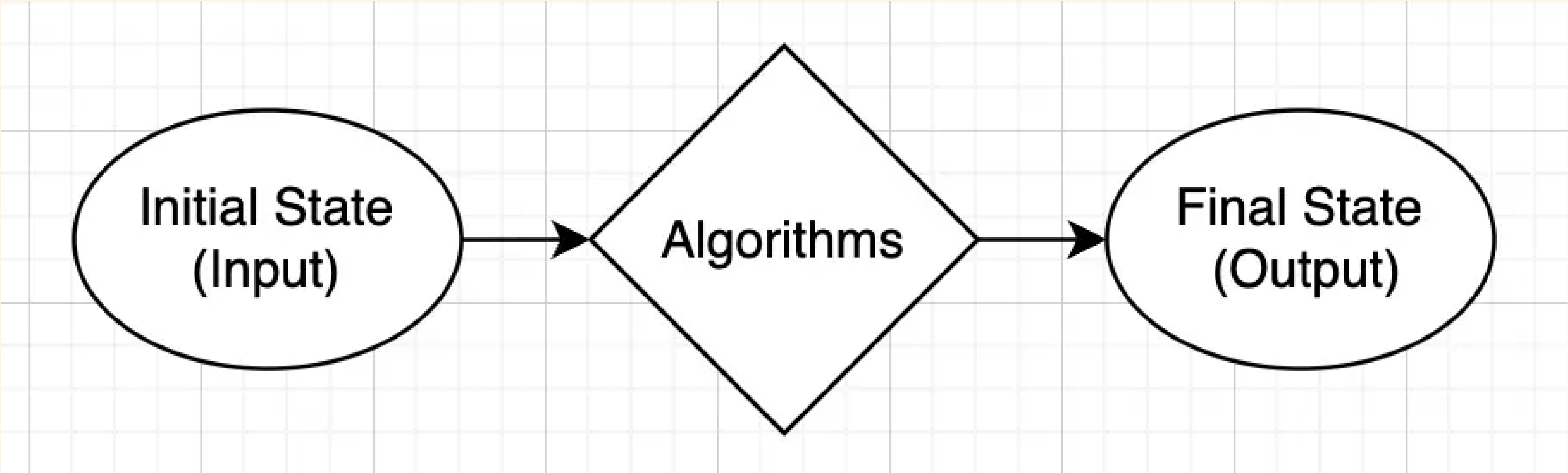
- Define the concept of an "algorithm".
- Explain the 5 key characteristics of an algorithm.
- Break down mathematical problems into algorithmic steps.
- Understand why we need Flowcharts.
- Have a first look at Flowgorithm, the tool we will use in this training.

02

What is Algorithm

What is an Algorithm?

When you hear the word "Algorithm", what is the first thing that comes to mind?



Algorithms are Everywhere!

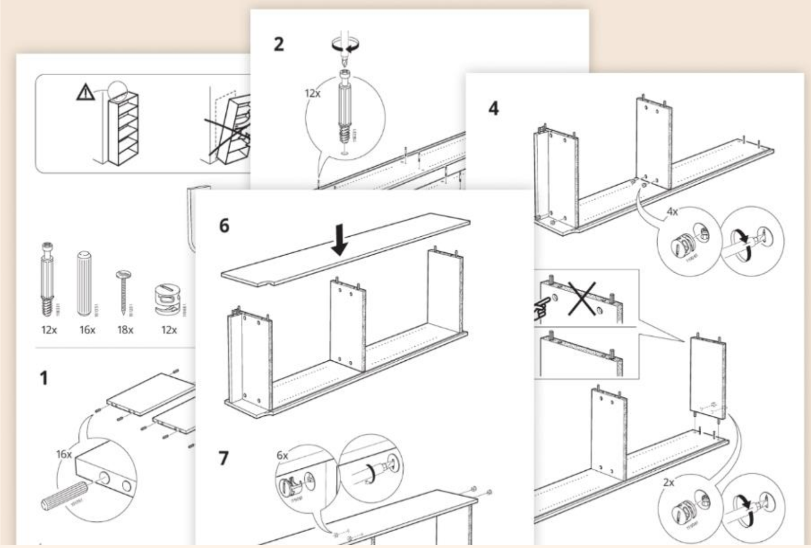
An algorithm is simply a clear, ordered set of steps to complete a specific task.

Real-Life: Recipe

A cooking recipe is sequential, has clear steps, and an output.

Algorithms are Everywhere!

An algorithm is simply a clear, ordered set of steps to complete a specific task.

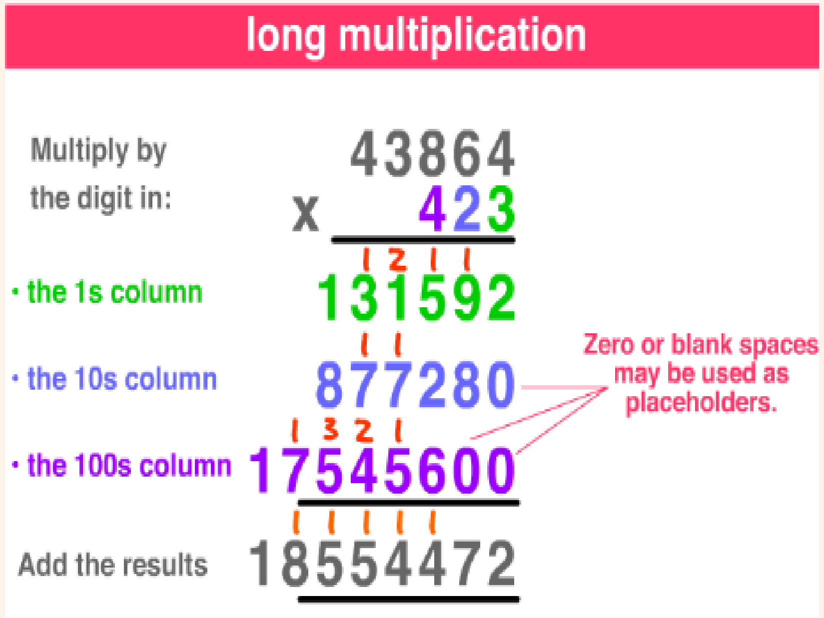


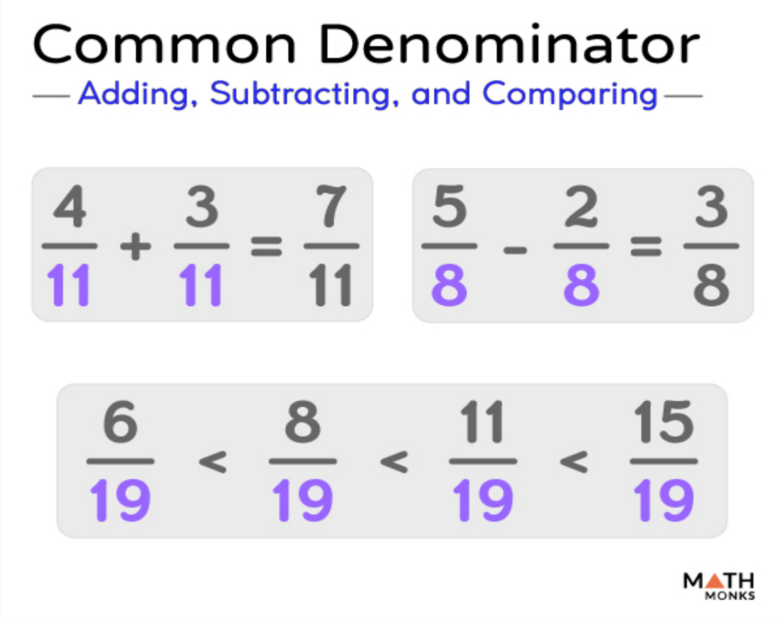
Real-Life: Assembly

An assembly guide for furniture follows a logical order.

Algorithms are Everywhere!

An algorithm is simply a clear, ordered set of steps to complete a specific task.





Mathematics

The steps for adding two fractions: find common denominator, add numerators, simplify.

Al-Khwarizmi (9th Century)

Definition & Origin

Today's Definition: A clearly defined, sequential, and finite set of steps designed to solve a problem.

Origin: The term comes from the name of the 9th-century Persian mathematician, Al-Khwarizmi.

His work was translated as "Algoritmi de numero Indorum" (Algorithms on Indian Numbers).

For centuries, "algorithm" referred to the "Arabic decimal system" before evolving to mean a set of calculation rules.

For us: Writing down the logical path from our minds, so a computer can understand it.

Why Mathematics and Algorithms?

Structures Thinking

Helps students break down problems into manageable steps.

Provides Clarity

Makes mathematical logic visual and structural.

Universal Language

A universal language, independent of any specific programming language.

Automation

Gives students the creative power to solve problems automatically with technology.

03

5 Must-Haves

The 5 "Must-Haves" of an Algorithm

For a set of instructions to be called an "algorithm", it must have 5 key characteristics:

1. Input
2. Process
3. Output
4. Finiteness
5. Definiteness

1. Input

The data that the algorithm receives from the outside to work on. An algorithm can have zero or more inputs.

Addition: Inputs = Number 1 , Number 2 .

Quadratic Equation: Inputs = coefficients a , b , c .

Factorial: Input = The number N .

2. Process

This is the "brain" of the algorithm. These are the logical or arithmetic steps it performs on the inputs to reach a result.

Addition: Sum = Number1 + Number2.

Equation: Delta = b*b – 4*a*c.

Comparison: If Number > 0 then...

3. Output

The result produced by the algorithm. Every algorithm must have at least one output.

Addition: Output = The value of Sum.

Equation: Output = Root 1 and Root 2.

Factorial: Output = The result of N!.

4. Finiteness

An algorithm must always terminate after a finite number of steps. A set of instructions that enters an infinite loop is not an algorithm.

✓ Algorithm (Finite)

Add all numbers from 1 to 100.

✗ Not an Algorithm (Infinite)

"Keep adding 1 forever" (without a stop command).

5. Definiteness

Every step must be clear, precise, and have only one meaning. No room for ambiguity.

Ambiguous (Not an Algorithm)

"Find a number close to the larger of two numbers."

Definite (Algorithm)

If Number1 > Number2, then Output Number1. Else, Output Number2.

04

Flowchart & Flowgorithm

From Thought to Visual: Flowcharts

Algorithm

The textual, sequential steps of the solution (Step 1, Step 2...).

→

Flowchart

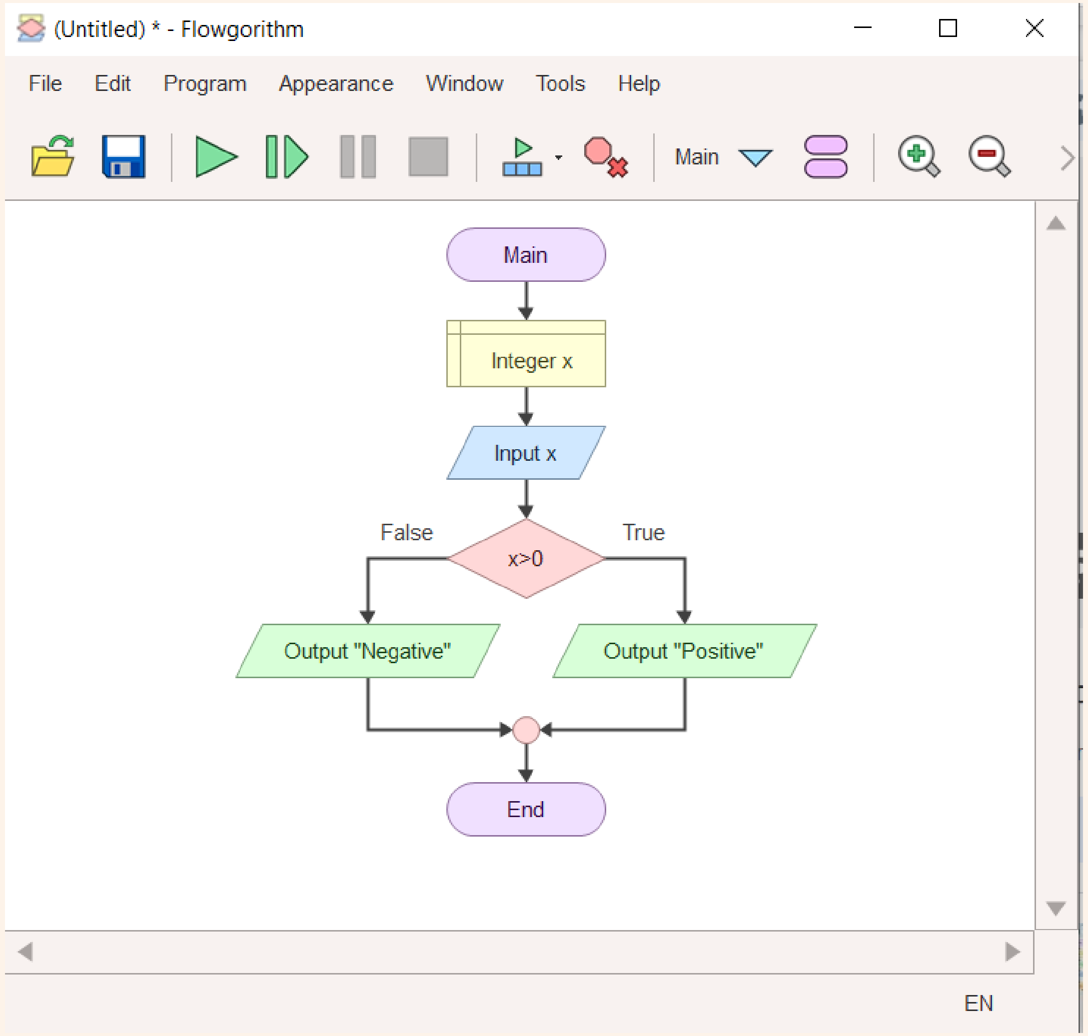
A visual diagram that shows the steps and flow using standard symbols.

Why do we use them?

Easier to understand (Visual)

Simpler to find errors (Debug)

A universal language



Our Tool: Flowgorithm

We will use a free educational software called Flowgorithm to draw and test our flowcharts.

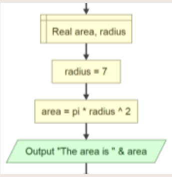
Why Flowgorithm?

- Uses standard flowchart symbols.
- Forces correct syntax while drawing.
- You can run the diagram and see the result.
- Translates your diagram into many languages, like Python, C++,C#, Javascript …

The Language of Flowgorithm

To create an algorithm, we drag and drop basic shapes between the "Main" and "End" ovals.

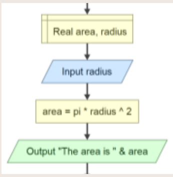
1. Start / End



2. Declare

3. Input

4. Process

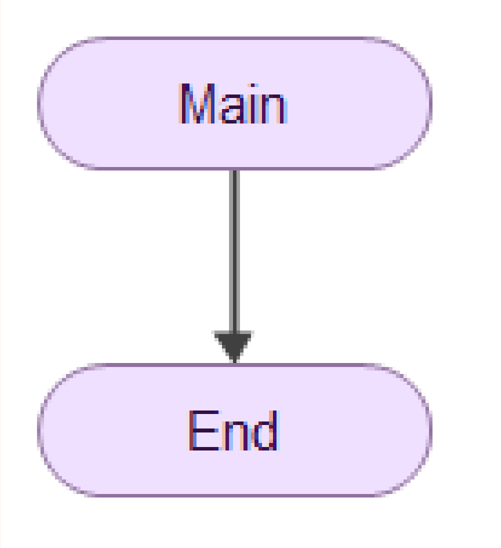


5. Output

(We will see the Decision and Loop symbols in the next sessions)

05

Symbols



Symbol 1: Start / End

Also known as the Terminator, it shows where the algorithm begins and ends.

In Flowgorithm, every flowchart automatically comes with a Main (Start) and End symbol.

START (Main) … … END

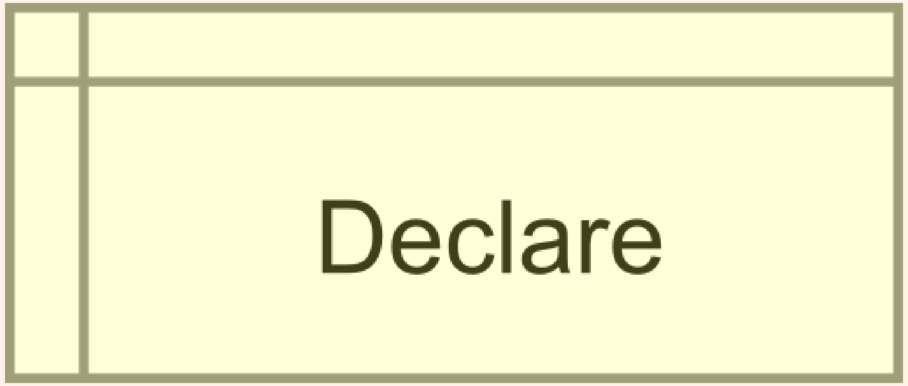
Symbol 2: Declare

VERY IMPORTANT: Flowgorithm requires you to "declare" (define) all your "memory boxes" (variables) at the very beginning.

What is a variable? A named memory location that holds information (a number, text, etc.).

Mathematical Equivalent: Our unknowns, like `x`, `y`, `Sum`.

When you click this shape, it asks for a variable name (e.g., `number1`) and a type (e.g., `Integer`).



Example: Declaring Variables

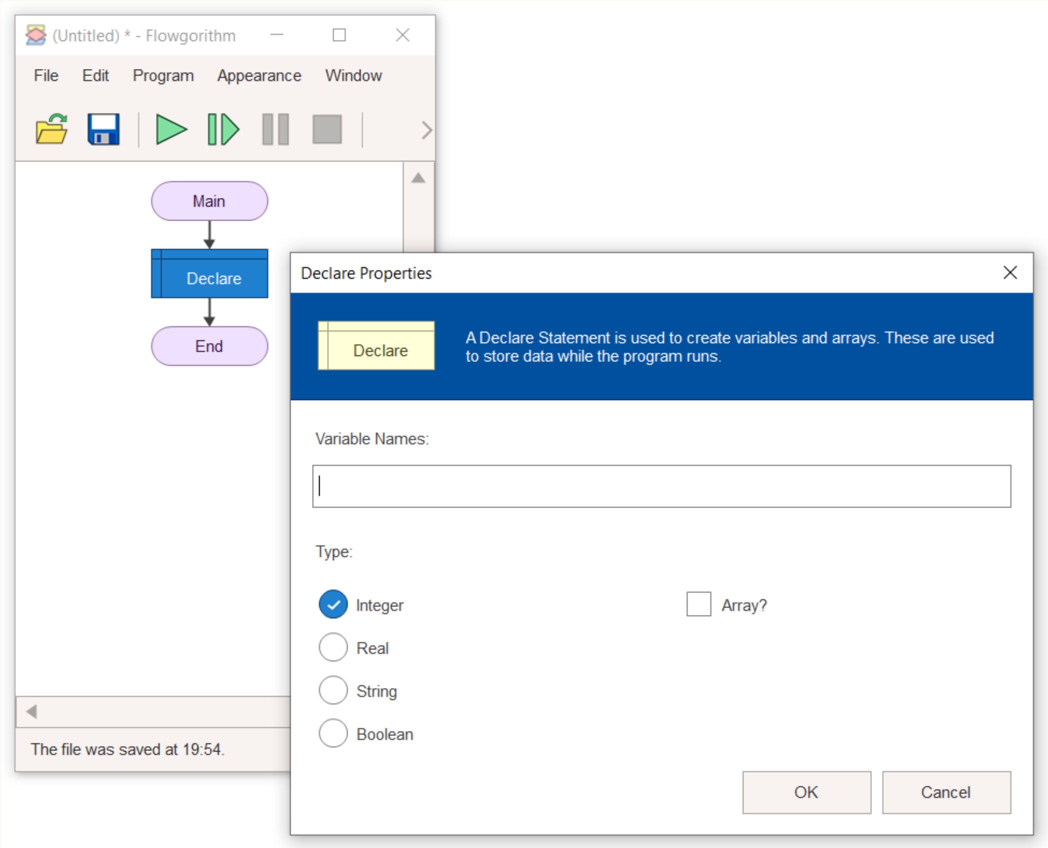
Problem: To add two integer numbers. We need places to store the first number, the second number, and the final sum.

Variable Name: `number1`, Type: `Integer`

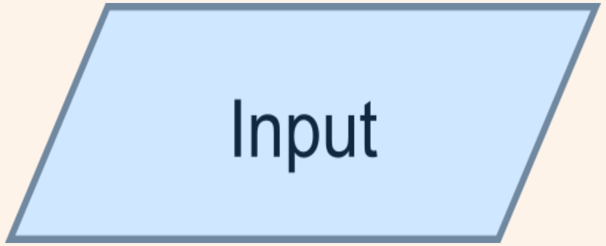
Variable Name: `number2`, Type: `Integer`

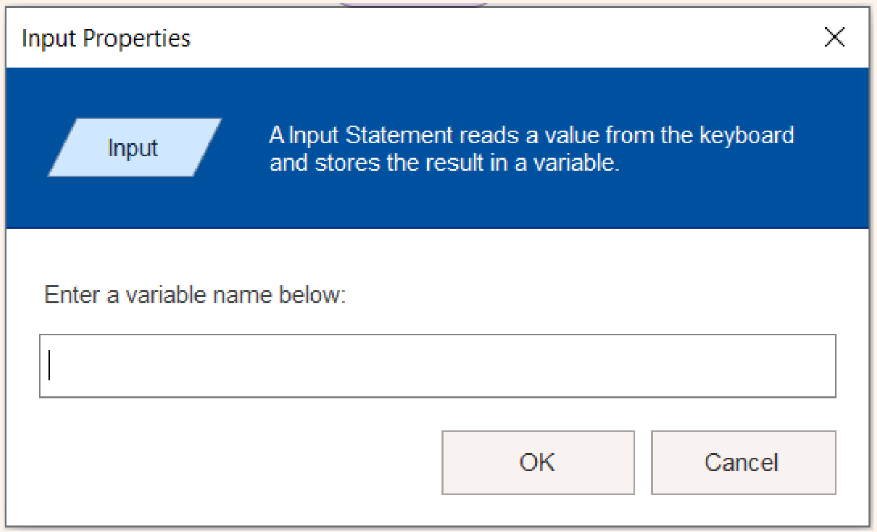
Variable Name: `sum`, Type: `Integer`

Memory Boxes



Note: We have not assigned values yet. We have only reserved the space in memory.





Symbol 3: Input

Used to get data from the user. This symbol fulfills the "Input" characteristic of an algorithm.

When you click this shape, it asks which variable you want to store the input in.

Example:
Input shape -> `number1` (The user enters a number, stored in `number1`)
Input shape -> `number2` (The user enters a second number, stored in `number2`)

Symbol 4: Process Symbols

This is where all mathematical calculations and value assignments happen. It fulfills the "Process" characteristic.

When you click this shape, it expects you to make an "Assignment".

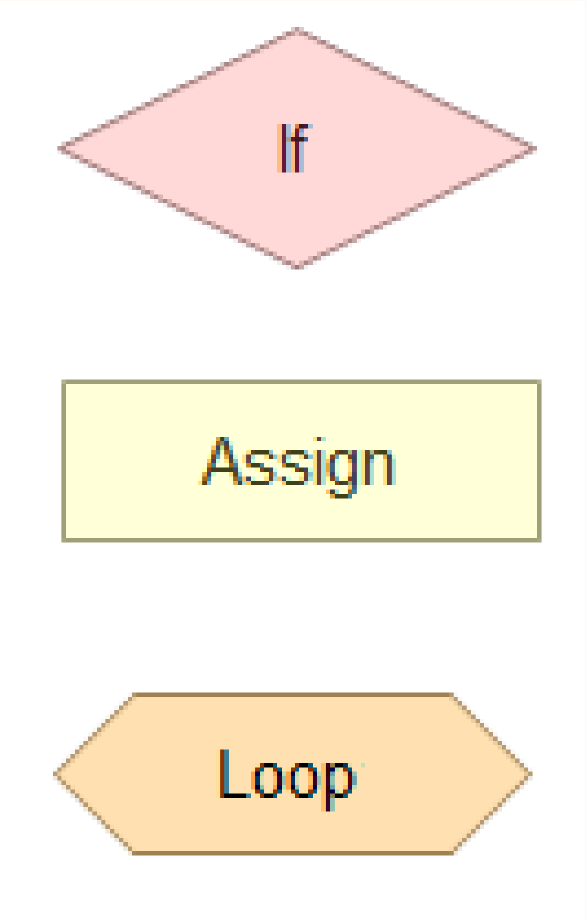
Format:

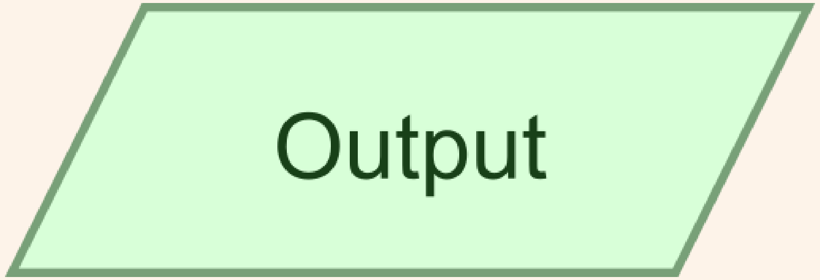
Variable = Mathematical Expression

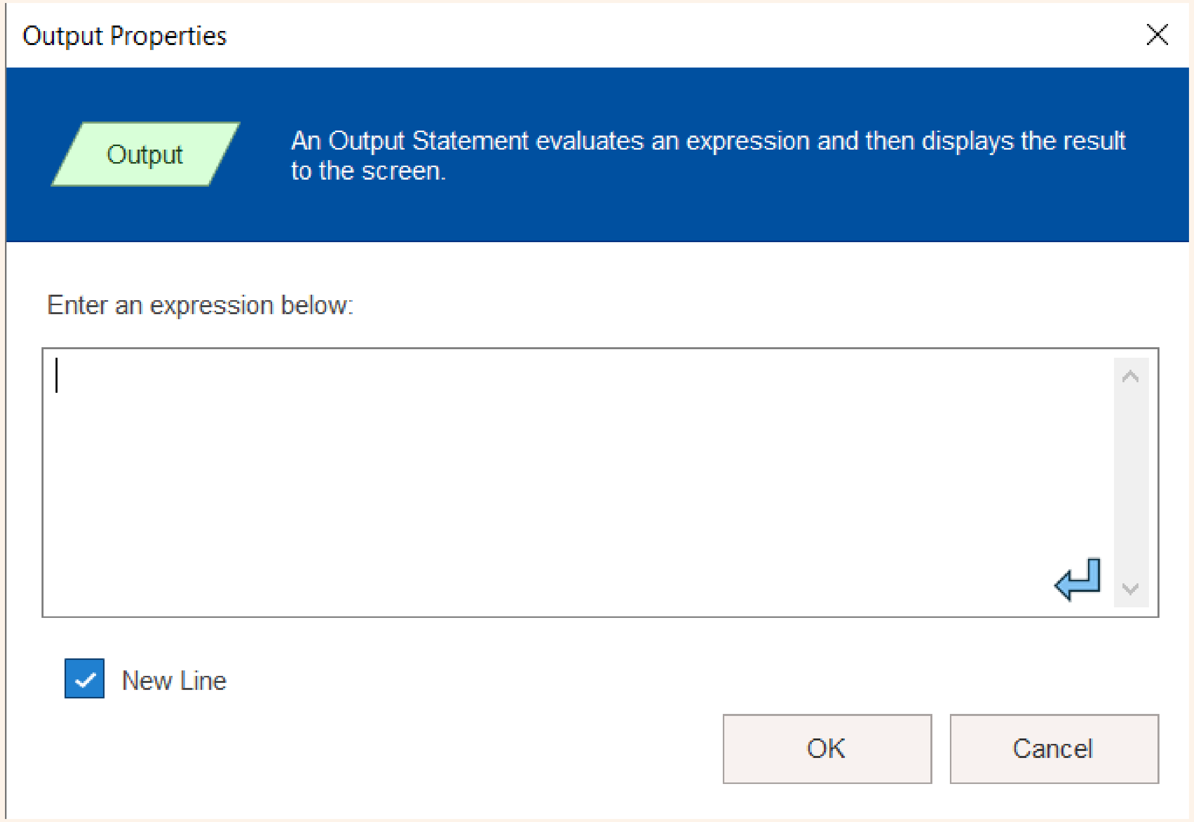
Example:

`sum = number1 + number2`

`delta = b*b - 4*a*c`







Symbol 5: Output

Used to display a result or a message to the user. This symbol fulfills the "Output" characteristic.

When you click this shape, it asks what you want to display.

Example:

Output shape -> sum (Displays the value of sum)

Output shape -> "The result is: " & sum (Displays combined text and value)

06

Live Examples

Example 1: Sum of Two Numbers

Problem: Write an algorithm and draw the flowchart to find the sum of two integers provided by the user.

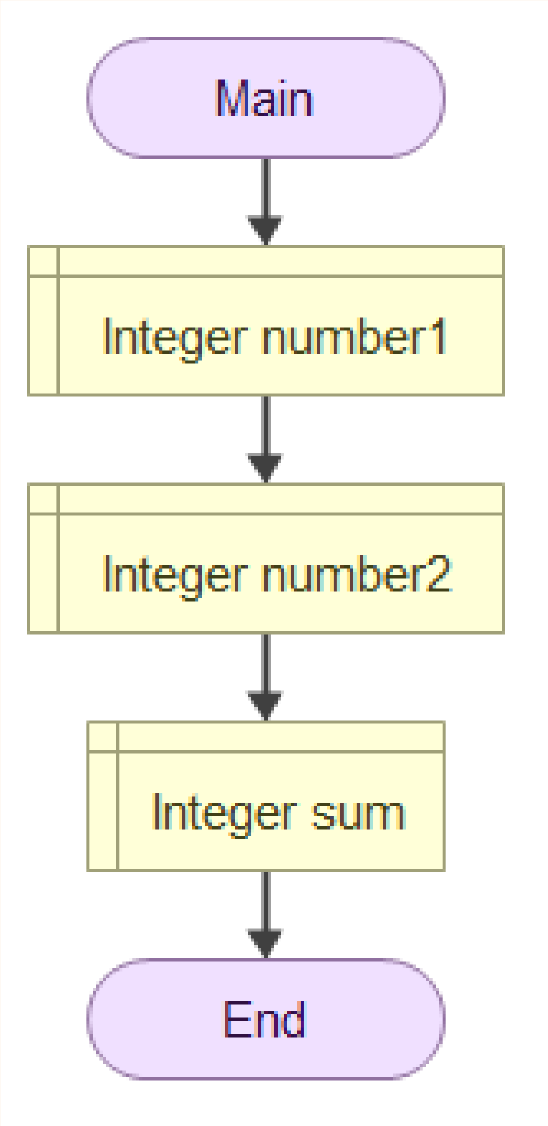
Textual Algorithm Representation (Pseudo Code):

1. Start.
2. Declare an **integer** variable named `**number1**`.
3. Declare an **integer** variable named `**number2**`.
4. Declare an **integer** variable named `**sum**`.
5. Get the value for `**number1**` from the user.
6. Get the value for `**number2**` from the user.
7. Process the operation: `**sum = number1 + number2**`.
8. Output the value of `**sum**` to the screen.
9. Stop.

Note: This text meets all 5 characteristics (Input, Process, Output, Finiteness, Definiteness).

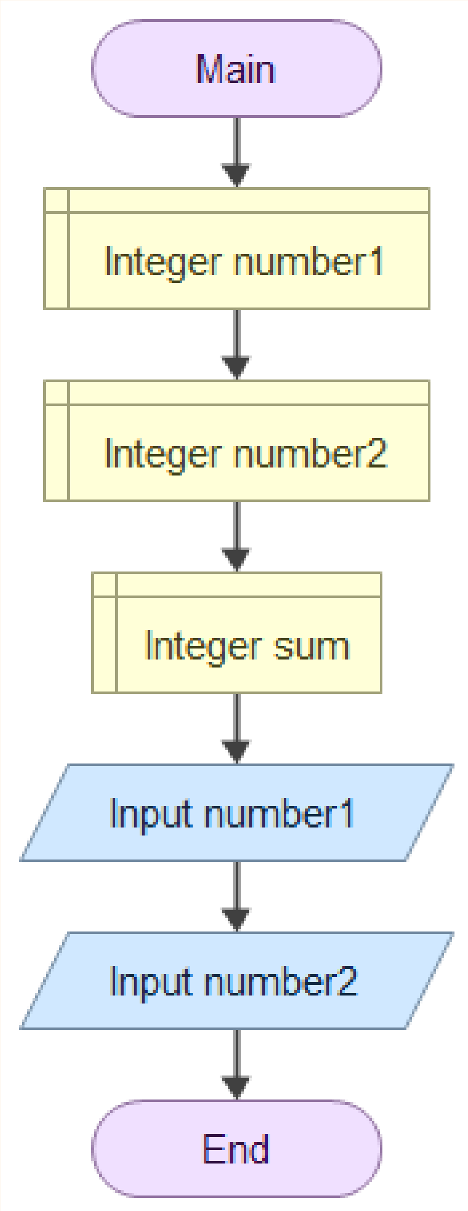
Ex1: Flowgorithm (Step 1: Declare)

1. Start.
2. Declare an **integer** variable named `**number1**`.
3. Declare an **integer** variable named `**number2**`.
4. Declare an **integer** variable named `**sum**`.



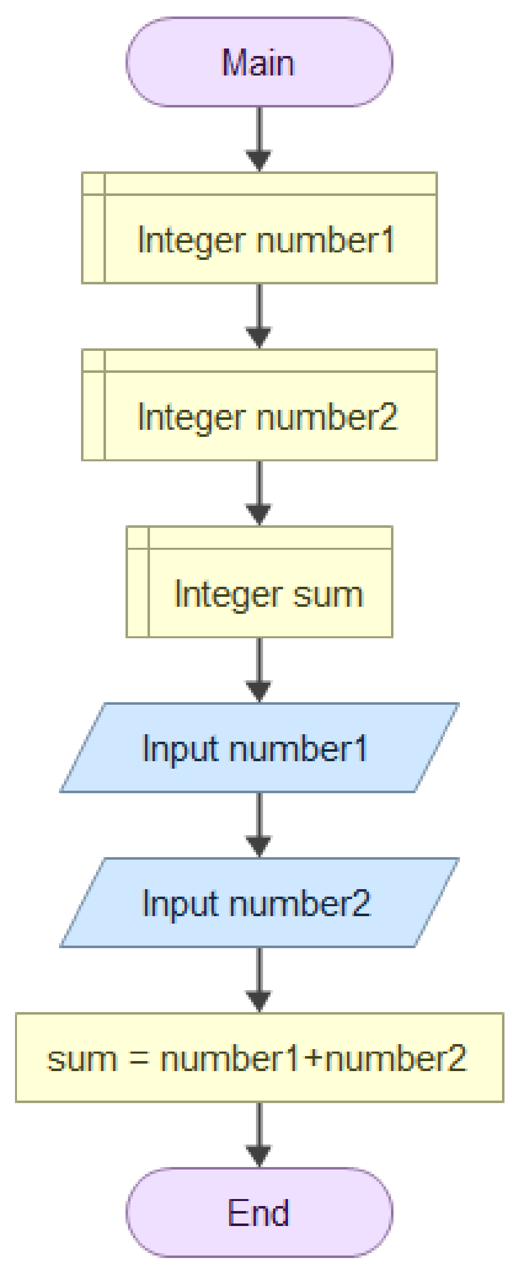
Ex1: Flowgorithm (Step 2: Input)

1. Start.
2. Declare an **integer** variable named `**number1**`.
3. Declare an **integer** variable named `**number2**`.
4. Declare an **integer** variable named `**sum**`.
5. Get the value for `**number1**` from the user.
6. Get the value for `**number2**` from the user.



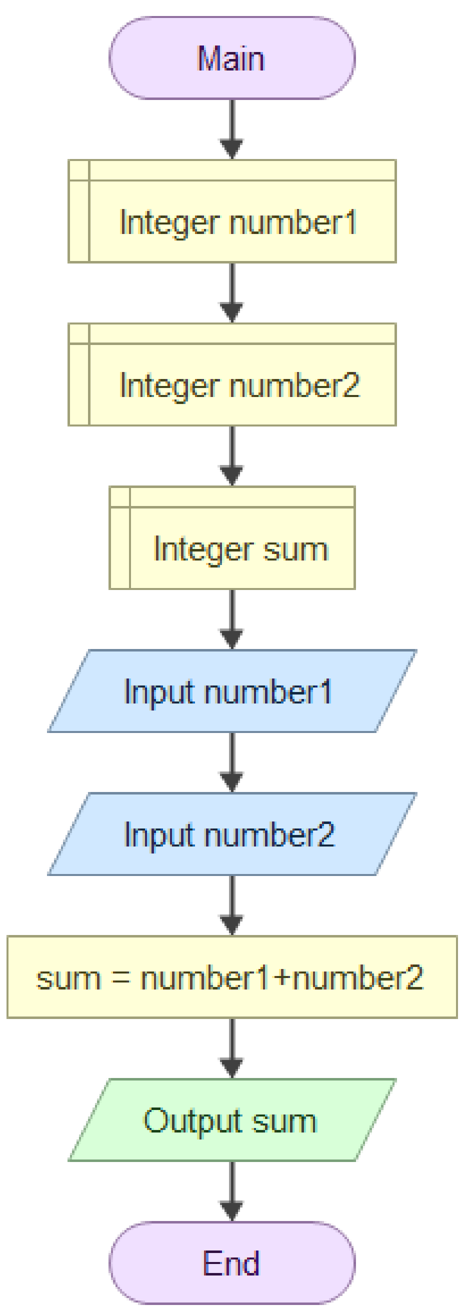
Ex1: Flowgorithm (Step 3: Process)

1. Start.
2. Declare an **integer** variable named `**number1**`.
3. Declare an **integer** variable named `**number2**`.
4. Declare an **integer** variable named `**sum**`.
5. Get the value for `**number1**` from the user.
6. Get the value for `**number2**` from the user.
7. Process the operation: `**sum = number1 + number2**`.

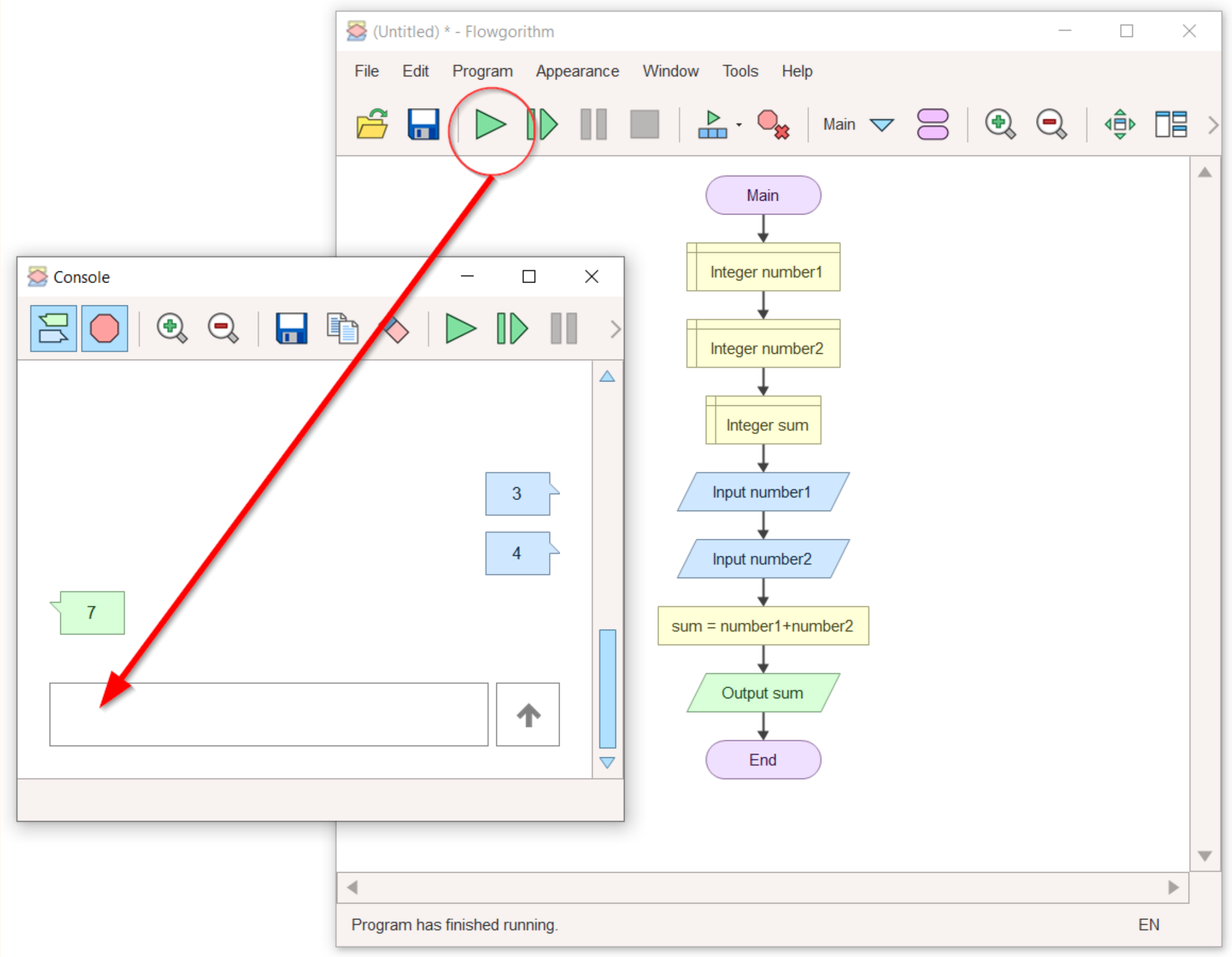


Ex1: Flowgorithm (Step 4: Output)

1. Start.
2. Declare an **integer** variable named `**number1**`.
3. Declare an **integer** variable named `**number2**`.
4. Declare an **integer** variable named `**sum**`.
5. Get the value for `**number1**` from the user.
6. Get the value for `**number2**` from the user.
7. Process the operation: `**sum = number1 + number2**`.
8. Output the value of `**sum**` to the screen.
9. Stop.



Running in Flowgorithm



Press the "Run" button (Green 'Play' icon) to follow the diagram step-by-step.

Console Window:

> Please enter a value for `number1`: 3
> Please enter a value for `number2`: 4
> (It performs the process)
>> 7

This is the fastest way to test your algorithm's logic without writing any code!

Example 2: Division (Quotient and Remainder)

Problem: Design an algorithm that takes two integers (Dividend and Divisor) and finds the integer quotient and the remainder.

Mathematical Background:

Quotient = Dividend \ Divisor (The integer part)

Remainder = Dividend % Divisor

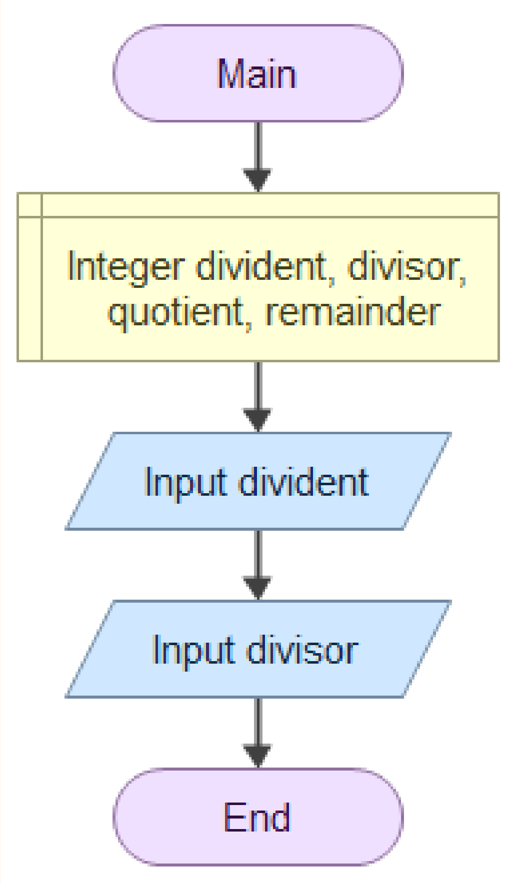
Example: 17 / 5 -> Quotient = 3, Remainder = 2.

Example 2: Textual Algorithm (Division)

1. Start.
2. Declare variables (Integer: `dividend`, `divisor`, `quotient`, `remainder`).
3. Get the `dividend` number from the user.
4. Get the `divisor` number from the user.
5. Process `quotient = Integer(dividend / divisor)`.
6. Process `remainder = dividend % divisor`.
7. Output "Quotient: " & `quotient`.
8. Output "Remainder: " & `remainder`.
9. Stop.

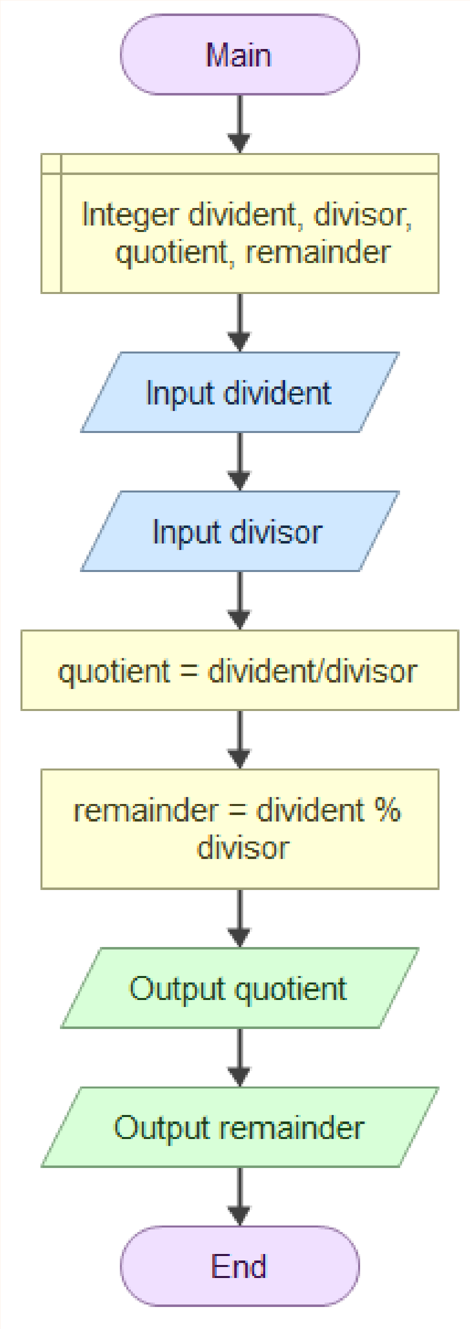
Ex2: Flowgorithm (Declare & Input)

1. Start.
2. Declare variables (Integer: `dividend`, `divisor`, `quotient`, `remainder`).
3. Get the `dividend` number from the user.
4. Get the `divisor` number from the user.
5. Process quotient = dividend / divisor.
6. Process remainder = dividend % divisor.
7. Output quotient
8. Output remainder
9. Stop.

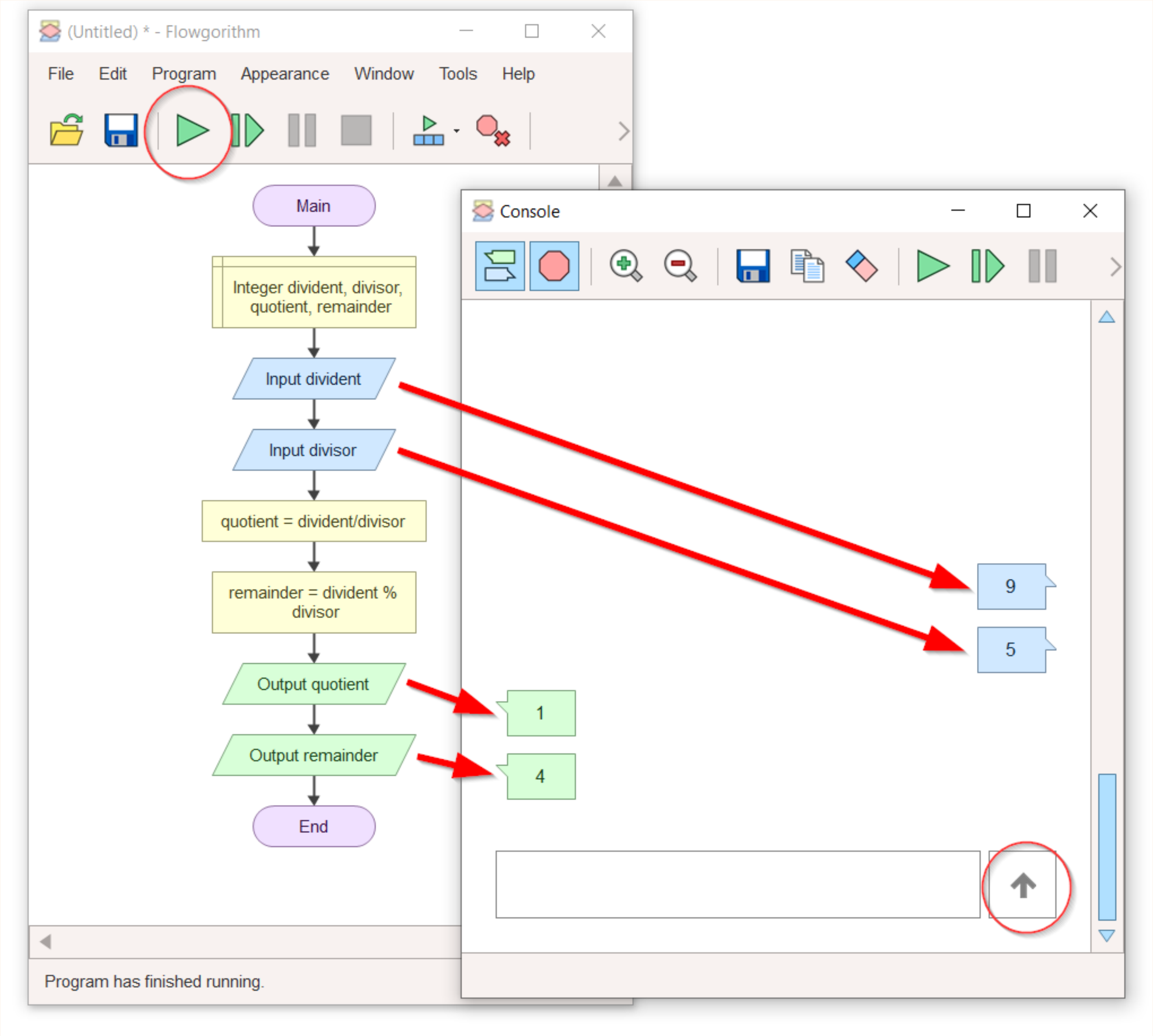


Ex2: Flowgorithm (Process & Output)

1. Start.
2. Declare variables (Integer: `dividend`, `divisor`, `quotient`, `remainder`).
3. Get the `dividend` number from the user.
4. Get the `divisor` number from the user.
5. Process `quotient = Integer(dividend / divisor)`.
6. Process `remainder = dividend % divisor`.
7. Output "Quotient: " & `quotient`.
8. Output "Remainder: " & `remainder`.
9. Stop.



Ex2: Completed Flowchart (Division)



07

Wrap-Up & Next

Session 1: Summary

Concepts

- **Algorithm:** Clear, sequential, finite steps.
- **5 Characteristics:** Input, Process, Output, Finiteness, Definiteness.
- **Flowchart:** A visual plan for an algorithm.

Flowgorithm Basics

- Declare: Creates variables.
- Input: Gets data from the user.
- Process: Performs calculations.
- Output: Displays the result.

What to Expect in Session 2

Flowchart Basics & Algorithmic Operators

- A detailed look at flowchart symbols.
- In-depth study of Assignment, Comparison, and Logical operators.
- Using the Decision symbol (If / Else) for complex logic.
- Drawing more complex mathematical flowcharts (e.g., Checking for Even/Odd).

Do you have any questions?

End of Session 1

Prof. Dr. Turgay Tugay BİLGİN
Bursa Technical University, 2025