

# Programming Fundamentals I

Chapter 1a and 1b: *Using Pseudocode and Flowcharts*

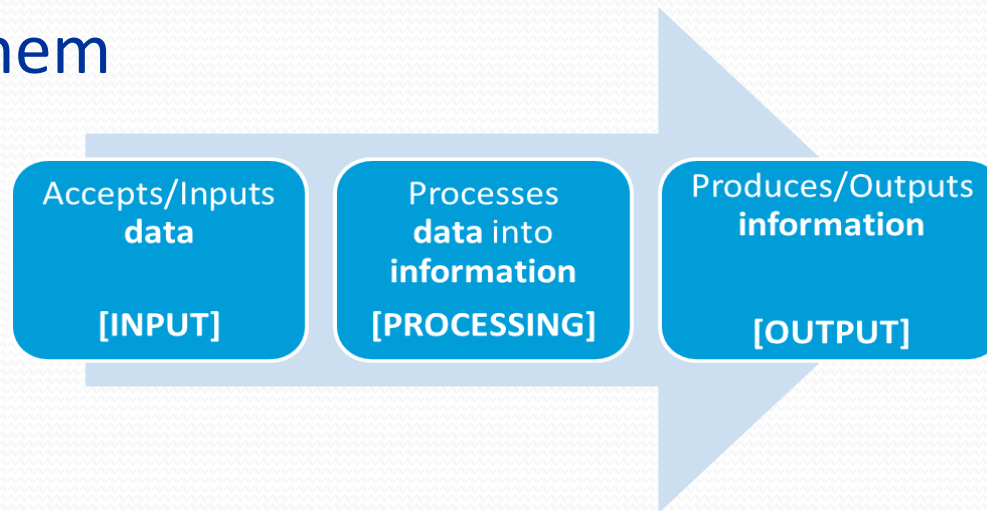
Dr. Adriana Badulescu

# Objectives

- Learn about pseudocode
- Learn about flowcharts
- Learn to solve problems and build algorithms and flowcharts
- Understand and use structures

# Computer Components and Programs

- A **computer** is an electronic device, operating under the control of instructions stored in its own memory
- A **program** or **software** consists of a series of instructions that tells the computer what tasks to perform and how to perform them



Information Processing Cycle

# Programming Logic

- The heart of the programming process lies in planning the program's logic
  - During this phase of the programming process, the programmer plans the steps of the program, deciding what steps to include and how to order them
  - An **algorithm** is the sequence of steps needed to solve any problem a finite amount of time
- The two most common planning tools are **flowcharts** and **pseudocode**

# Programming Logic

- Programmer should define the sequence of events that will lead to the desired output
- Planning a program's logic includes:
  - Thinking carefully about all the possible data values a program might encounter
  - How you want the program to handle each scenario
- The process of walking through a program's logic on paper before you actually write the program is called **desk-checking**

# Programming Logic

## FLOWCHART

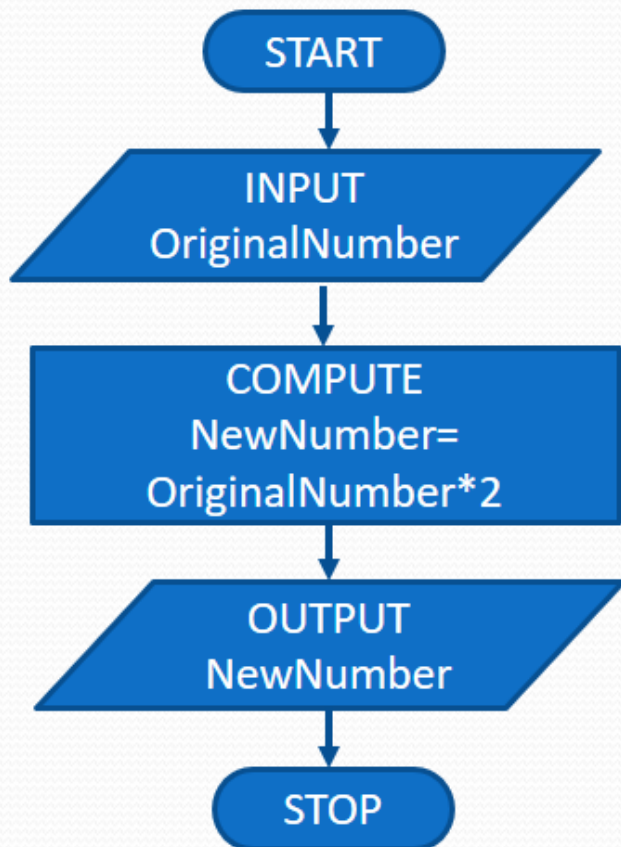
- A **flowchart** is a pictorial representation of the solution to a problem/algorithm

## PSEUDOCODE

- **Pseudocode** is an English-like representation of the logical steps it takes to solve a problem

# Programming Logic

## FLOWCHART



## PSEUDOCODE

START

INPUT OriginalNumber

COMPUTE NewNumber=  
OriginalNumber\*2

OUTPUT NewNumber

STOP

# Programming Logic

- Even if the pseudocode seems easier to write (especially for simple problems), when the problems get more complex and there are branches and repetition, if not carefully managed, can cause cases in which the algorithm/program will not end, thus, flowcharts are better because it is easier to spot the branches that do not end into the STOP symbol, thus, for this class, we are going to use flowcharts



# Using Flowchart Symbols


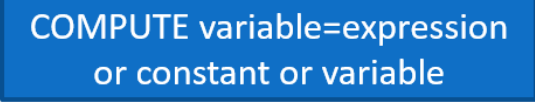







- A **flowchart** is a pictorial representation of the solution to a problem/algorithm/program
- When creating a **flowchart**, draw geometric shapes around individual statements and connect them with arrows

# Using Flowcharts

- How to store the computer knowledge

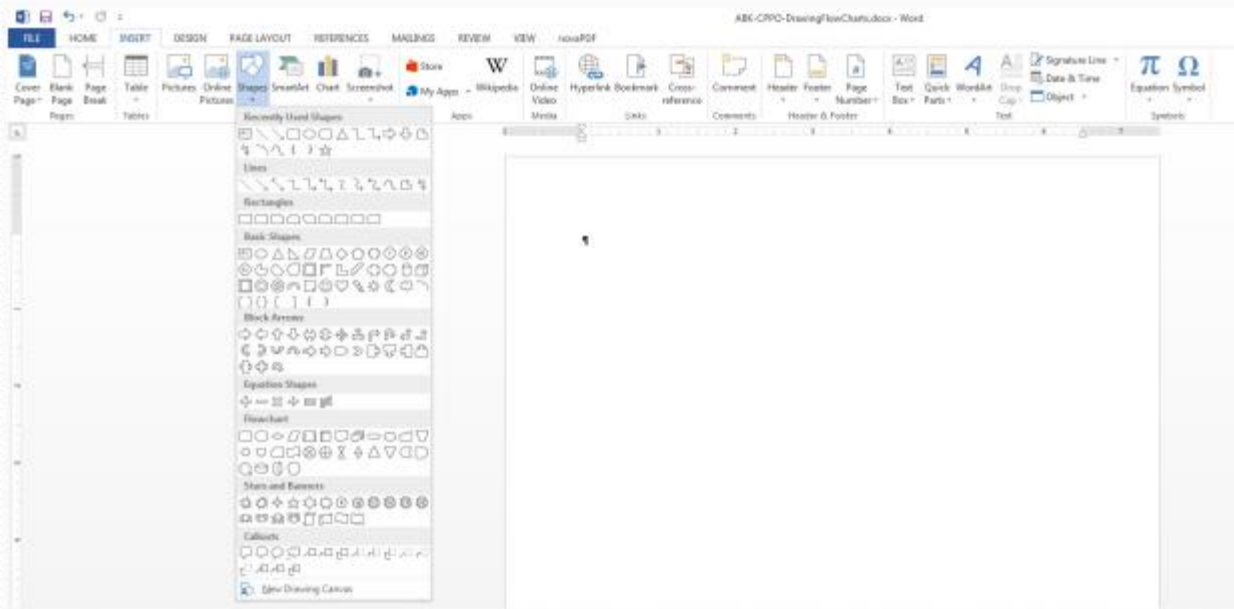
Memory	Variables	Constants	Expressions
Can Change?	Yes	No	No
Naming Convention	Contains letters, digits, and/or underscore (_) Should not start with a digit	<b>Numerical constants:</b> optional sign and digits <b>String/Textual constants:</b> sequence of character between double quotes ("")	Any combination of variables and constants using numerical or string operators/operations
Naming Example	Variable NewVariable X X1	10 -100 "X" "cat" "10"	-X X+1 3-9 x+y+6 "cats"+"&"+"dogs"

# Using Flowchart Symbols

Symbol	Shape	Meaning	Graphical Representation
<b>Input</b>	Parallelogram	Input operation	
<b>Processing</b>	Rectangle	Processing/calculation operation	
<b>Output</b>	Parallelogram	Output operation	
<b>Start</b>	Ellipses	The start of the algorithm	
<b>Stop</b>	Ellipses	The end of the algorithm	
<b>Decision</b>	Diamond	A decision/branch made based on question/condition	
<b>Connector</b>	Circle	A connector (entry to or exit from) an algorithm segment	
<b>Module</b>	Rectangle with double line	A predefined (named) process/module/subroutine defined in separate flowchart	
<b>Flowlines</b>	Arrow	Connect symbols into the correct sequence	

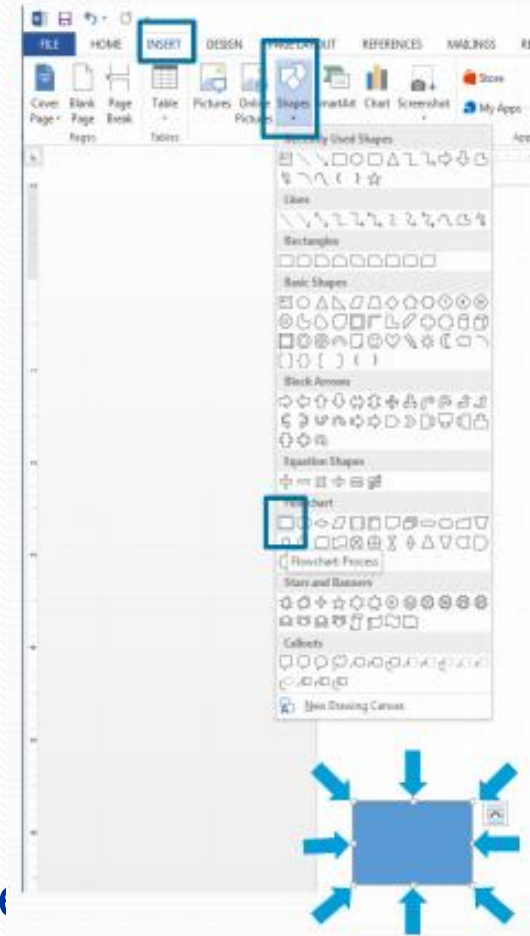
# Drawing Flowcharts in Microsoft Word Using Shapes

- Open INSERT Ribbon
- Select/Open Shapes (the down arrow)
- Select the shape from the Flowchart section



# Drawing Flowcharts in Microsoft Word Using Shapes

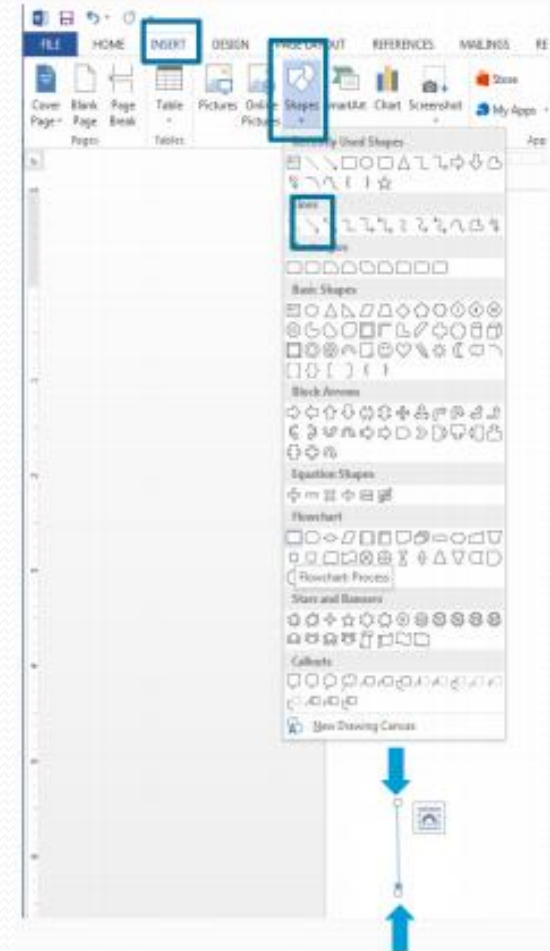
- Draw **Parallelogram, Rectangle or Ellipsis**
  - Click/Open INSERT Ribbon
  - Select/Open Shapes (the down arrow)
  - Select the shape from the Flowchart section. The mouse pointer becomes a cross
  - Click on the document to mark the beginning of the top-left corner and keep the mouse down and release when you reach the appropriate shape or click again to get a default size
  - Use the square points to resize it (see the blue arrows )



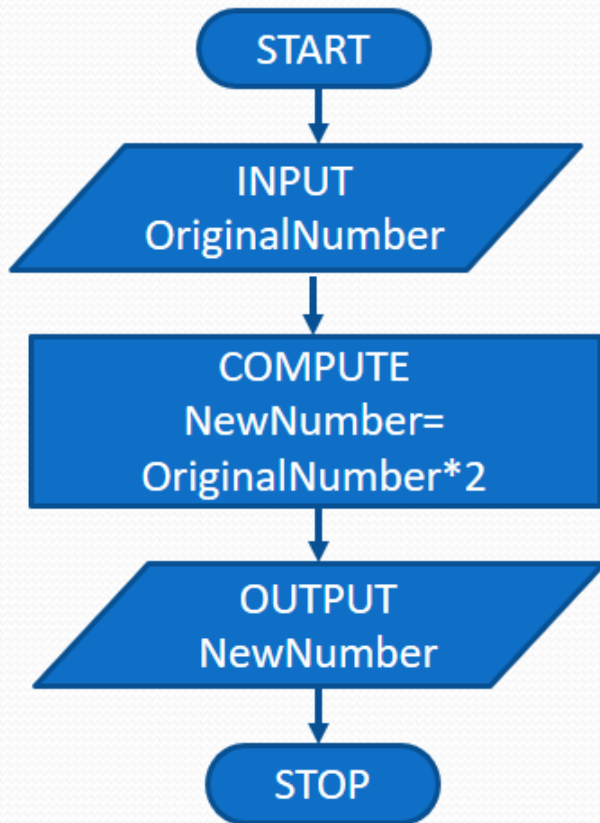
# Drawing Flowcharts in Microsoft Word Using Shapes

## ■ Draw Lines

- Click/Open INSERT Ribbon
- Select/Open Shapes (the down arrow)
- Select the arrow from the Flowchart section. The mouse pointer becomes a cross. Click on the document to mark the beginning of the arrow and keep the mouse down and release when you reach the end of the arrow or click again to get a default size
- Use the square points to resize it



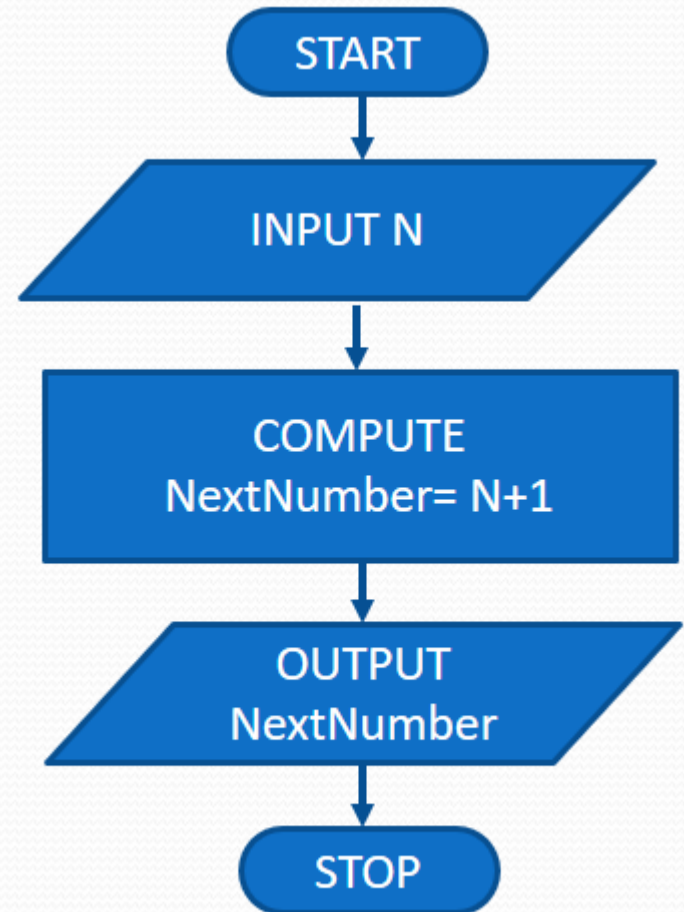
# Using Flowchart Symbols and Pseudocode Statements





# Next/Following Number



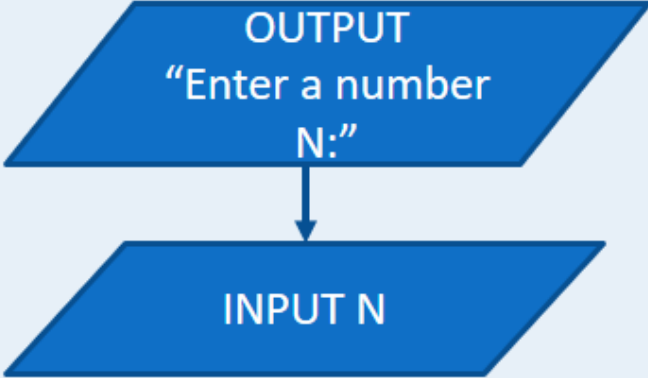
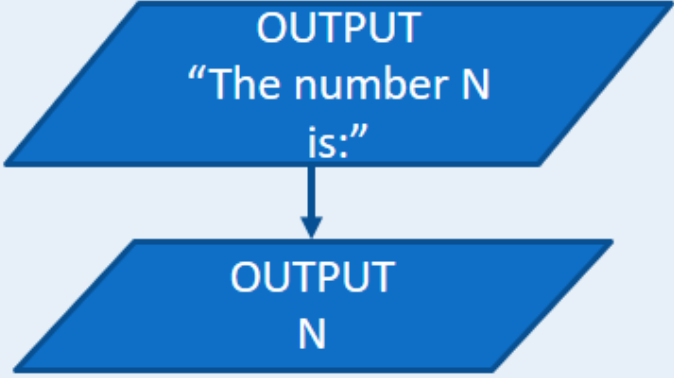
- Read a number  $N$  from the user and compute and output the next/following number (e.g. for 5 is 6, for -9 is -8)



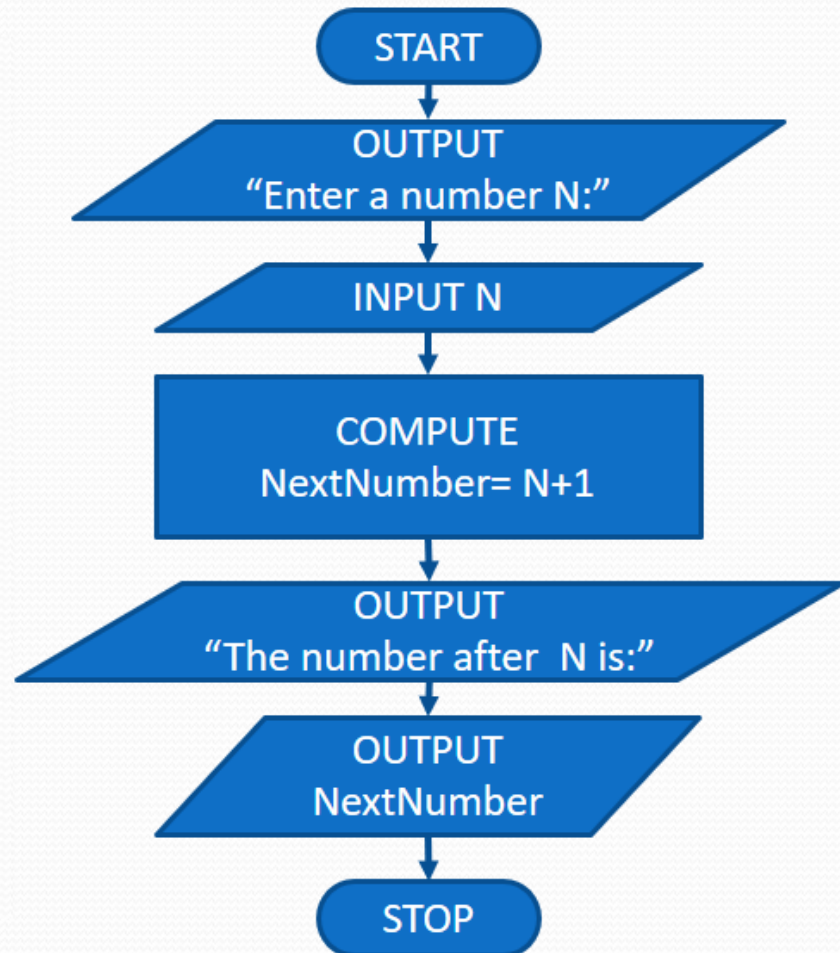
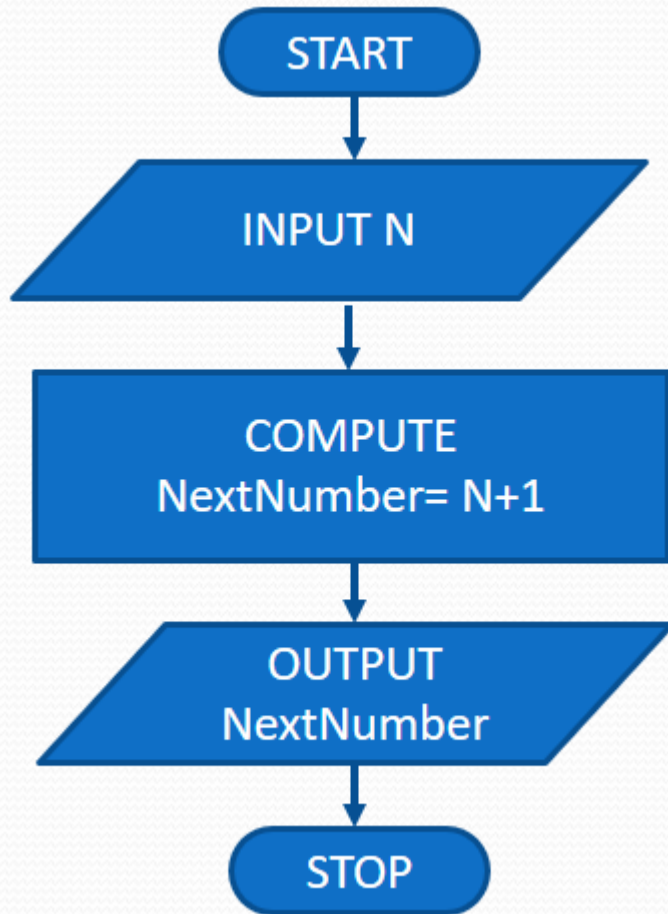


# User-friendly Flowcharts

- Your flowchart should be user friendly and show/give the user prompt messages that tells the user what he/she is supposed to enter or what the output means

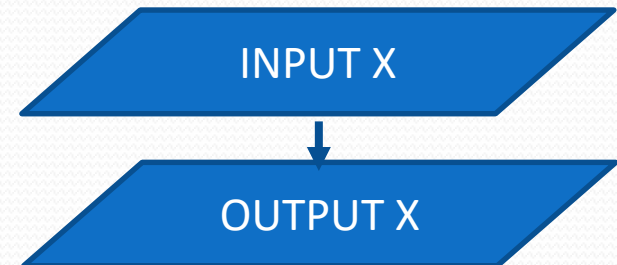
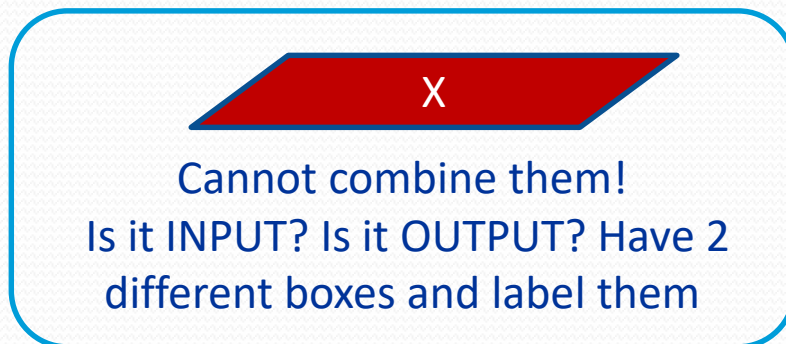
	INPUT	OUTPUT
Flowchart		
User-friendly Flowchart		

# User-friendly Flowcharts



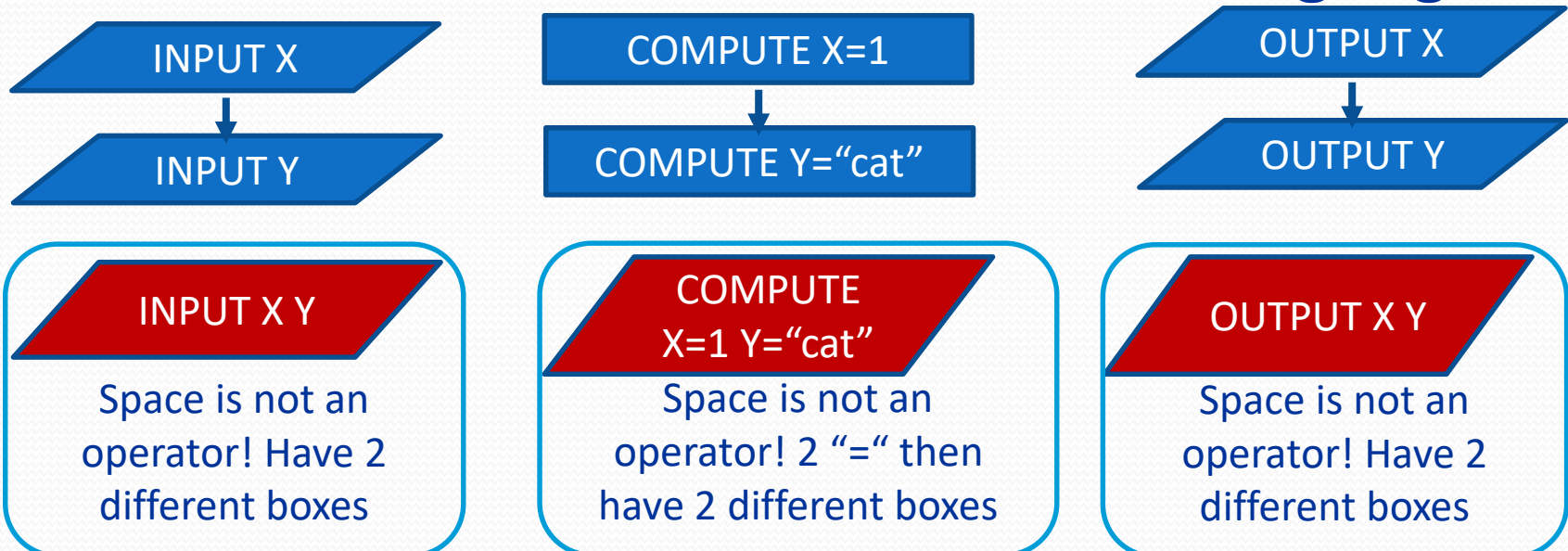
# Combining Flowchart Symbols

- You cannot combine different types of operation (different boxes for INPUT, for COMPUTE, for OUTPUT)
- E.g. If you need to INPUT x and then OUTPUT x, you cannot just combine them



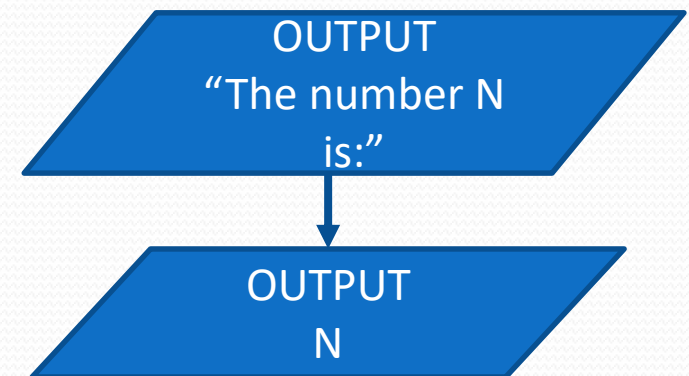
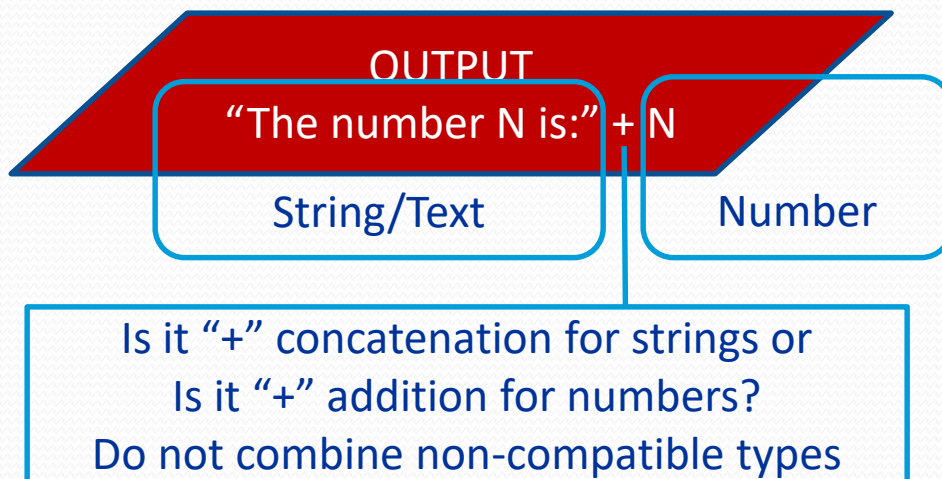
# Combining Flowchart Symbols

- You should not combine 2 separate operations even if they are of the same type because some languages cannot combine 2 operations and flow charts should work for all the languages



# Combining Flowchart Symbols

- Can I use an operator when the operands are the same?
- NO. You should not combine 2 separate operations even if they are of the same type since some languages cannot combine 2 operations



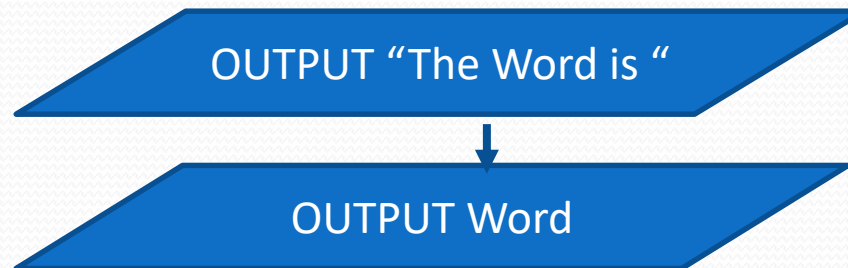
# Combining Flowchart Symbols

- But what if the two are of the same type
- Can I combine them using an operator?



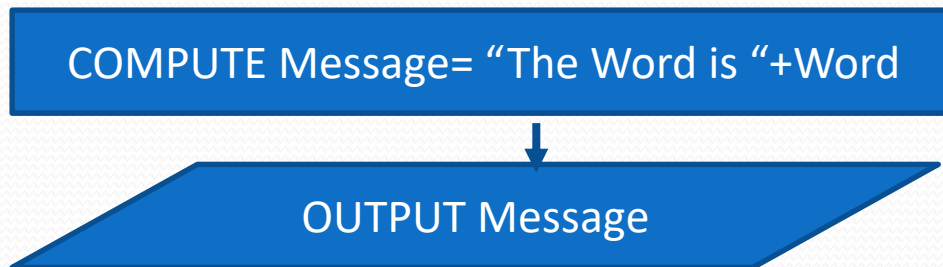
OUTPUT "The Word is "+Word

- Even if, some languages let you do that, some do not, and the flow charts needs to be general and work for any language, so NO.



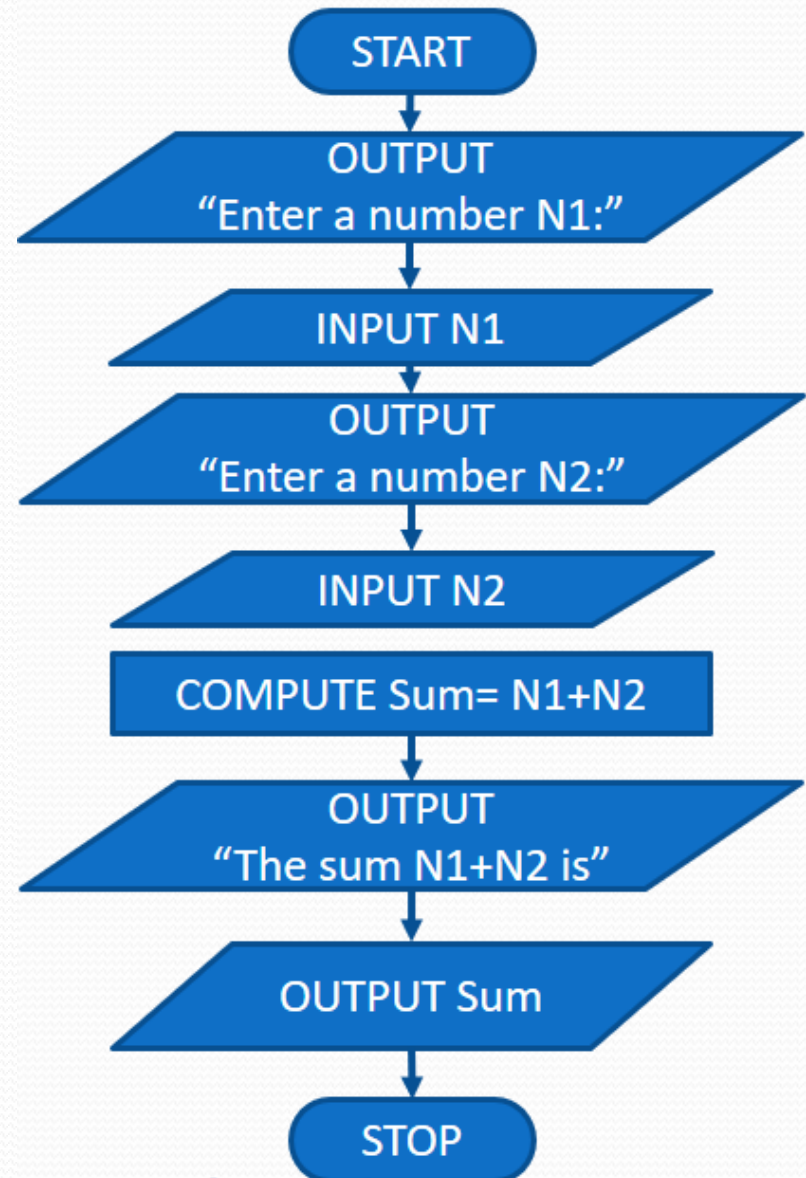
# Combining Flowchart Symbols

- Plus, when you use an operator, you are doing a calculation, so, you need to use a COMPUTE box for the operator, since again some languages do not allow calculations in the output



# Sum of 2 Numbers

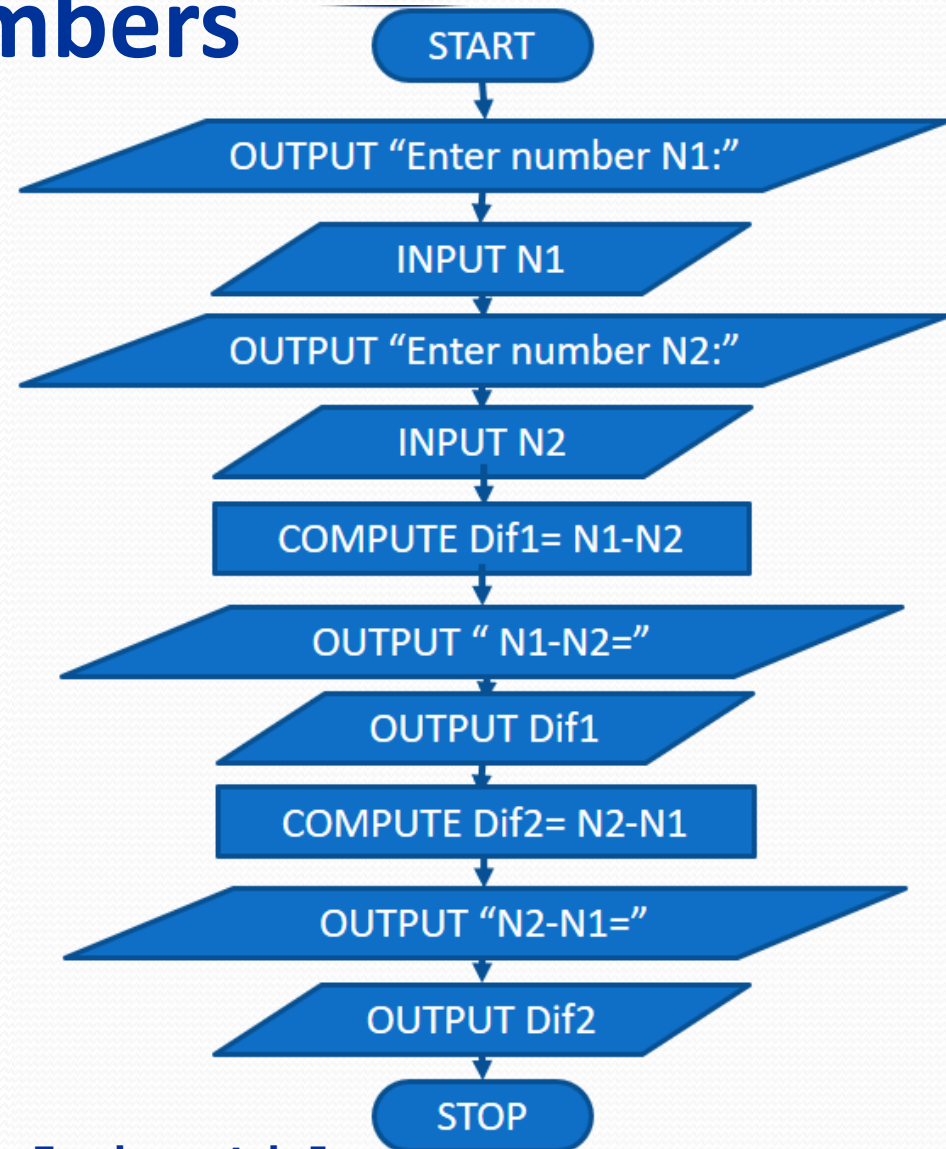
- Read 2 numbers N1 and N2 from the user and compute and output the sum of the 2 numbers





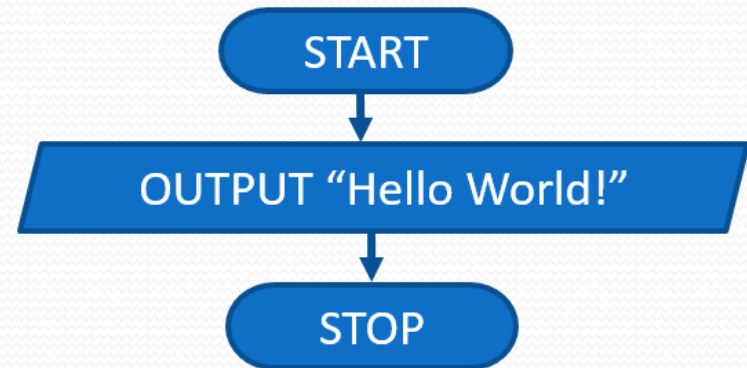
# Difference of 2 Numbers

- Read 2 numbers N1 and N2 from the user and compute and output the difference (N1-N2 and N2-N1) of the 2 numbers in one flowchart

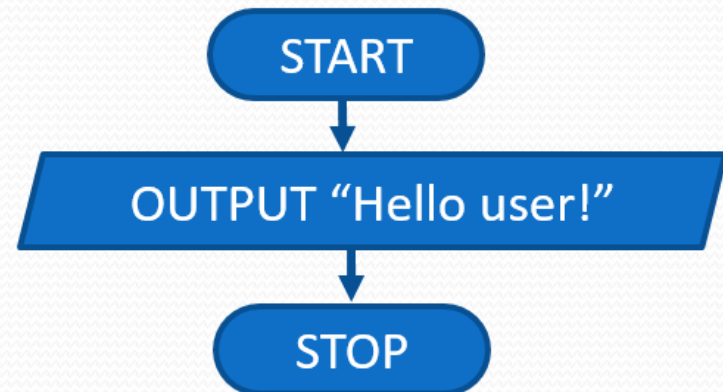


# Messages

- Say “Hello World!”

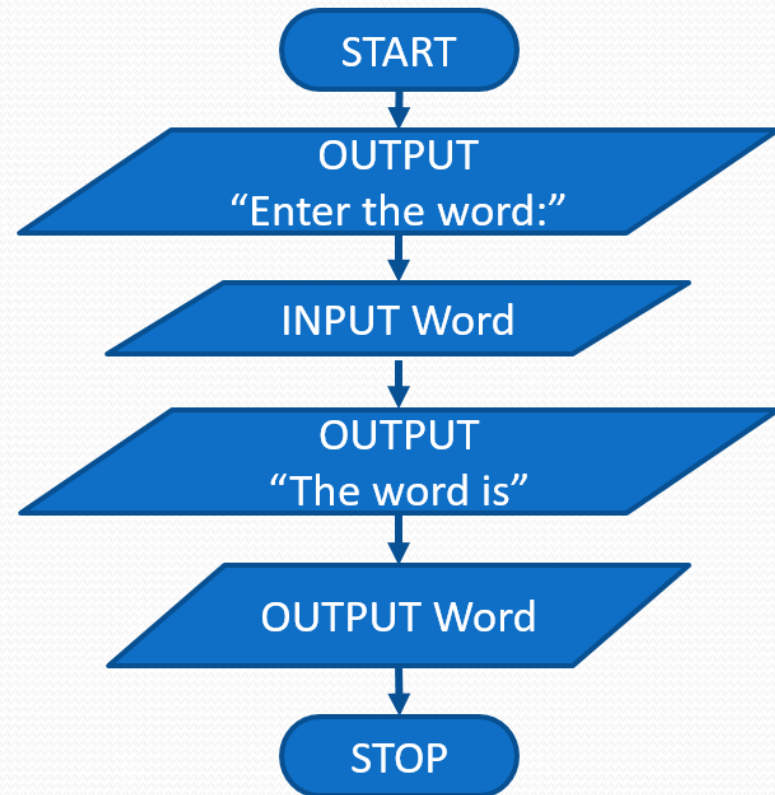


- Say “Hello user!”



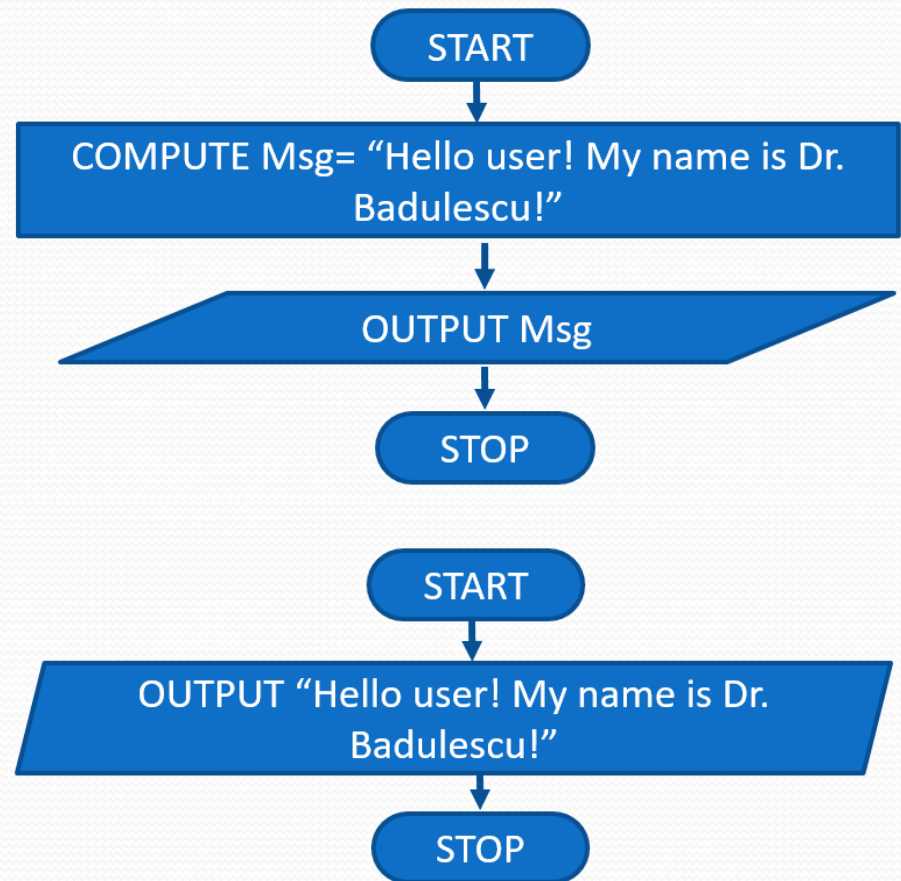
# Message

- Echo a word from the user



# Message 1

- Output the message: “Hello user! My name is [PROGRAMMER]!”. Replace [PROGRAMMER] with the programmer’s name

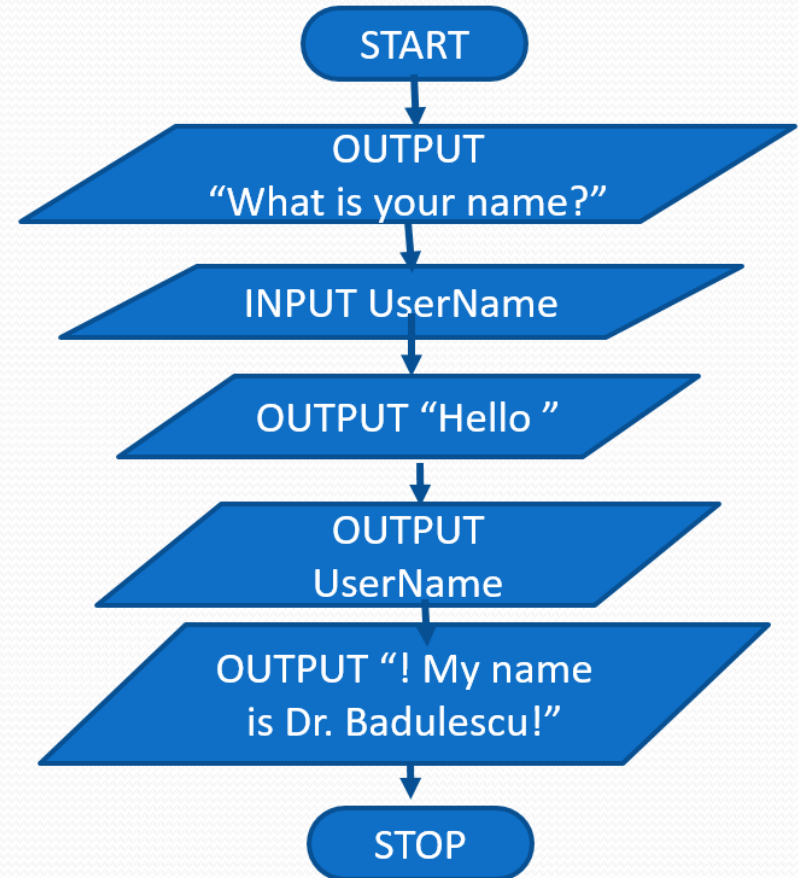
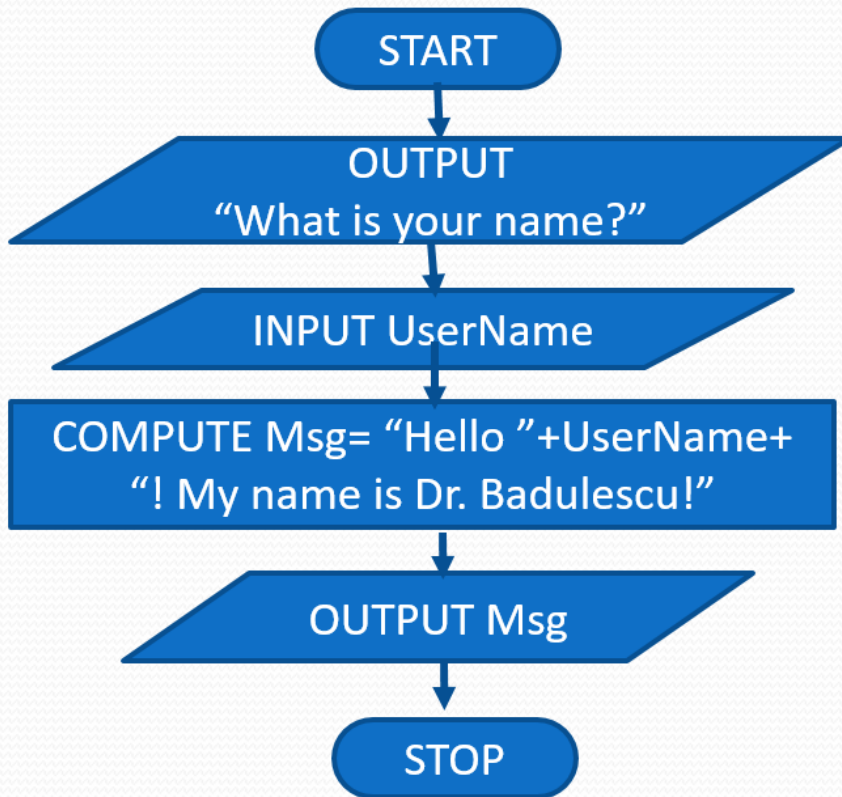


# Message 2

- Ask user for their name and output the message:  
“Hello [USER]! My name is [PROGRAMMER]!”.  
Replace [USER] with the user’s name and  
[PROGRAMMER] with the programmer’s name

H	e	l	l	o		[USER]	!		M	y		n	a	m	e		i	s		[PROGRAMMER]	!
---	---	---	---	---	--	--------	---	--	---	---	--	---	---	---	---	--	---	---	--	--------------	---

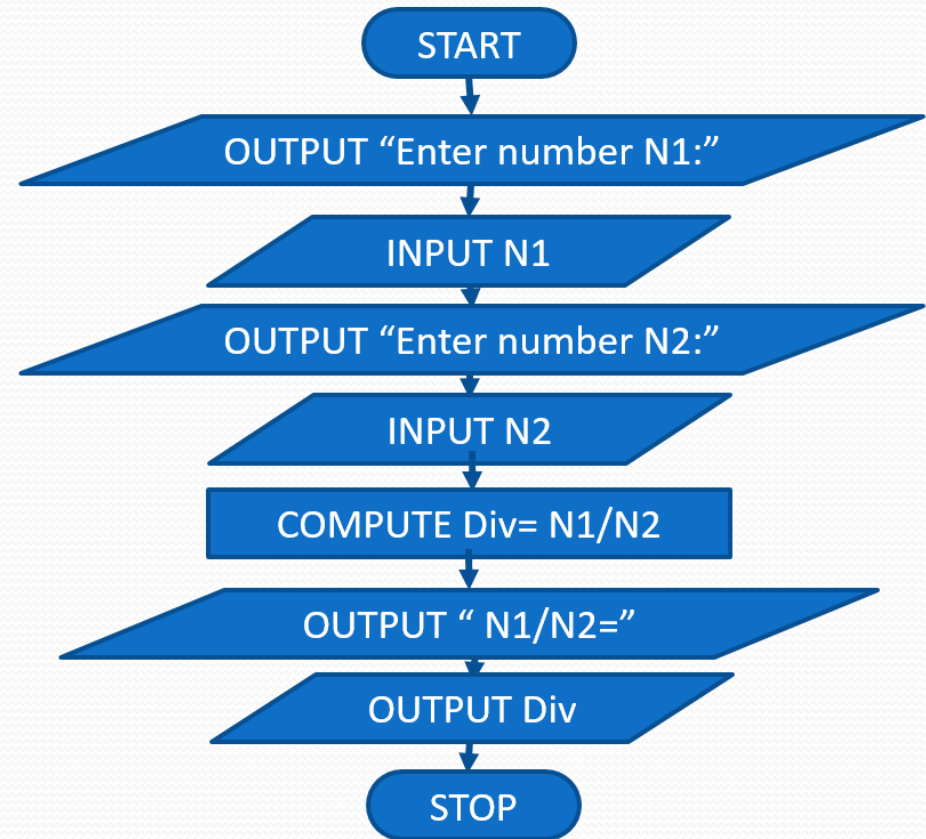
# Message 2





# Division of 2 Numbers

- Read 2 numbers N1 and N2 from the user and compute and output the division/quotient of the 2 numbers



Does not solve the problem since it does not work correctly if N2 is 0

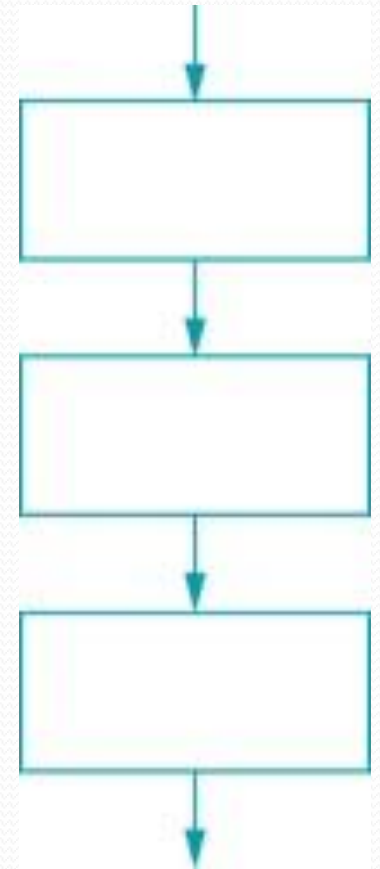
# Understanding the Three Basic Structures: Sequence, Selection, and Loop

- A **structure** is a basic unit of programming logic
  - Sequence
  - Selection
  - Repetition/Loop
- One can diagram each structure with a specific configuration of flowchart symbols



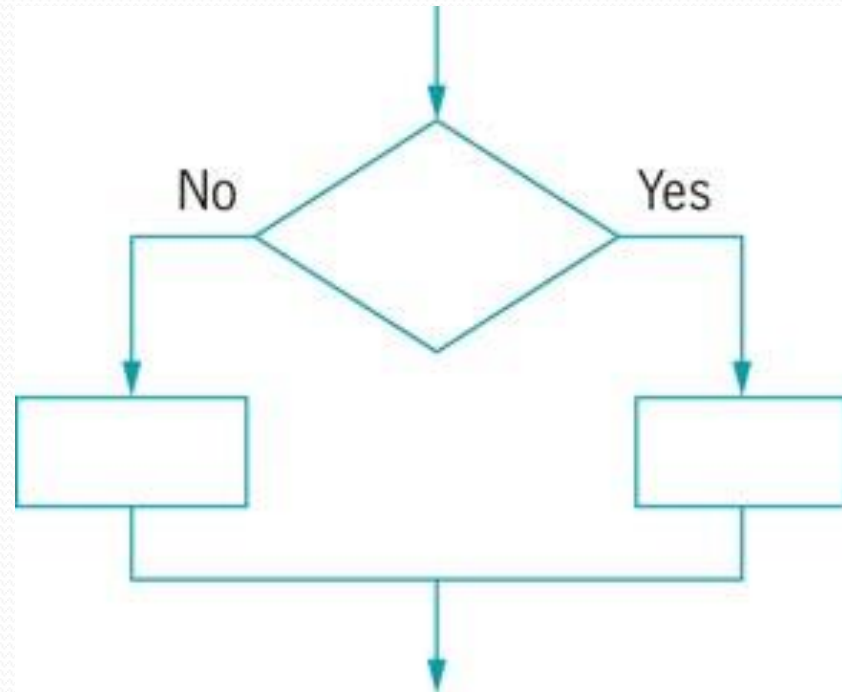
# The Sequence Structure

- Perform an action or task, and then perform the next action, in order
- Can contain any number of tasks
- No chance to branch off and skip any of the tasks
- Continue step-by-step until the sequence ends



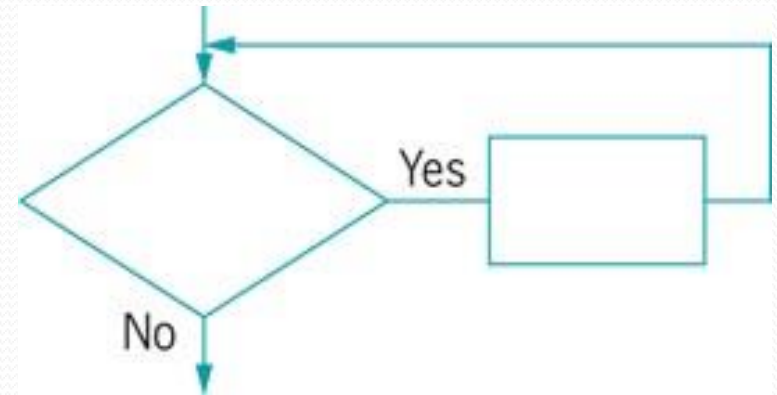
# The Selection Structure

- Ask a question and, depending on the answer, take one of two courses of action
- No matter which path followed, continue with the next task
- Also called a **if-then-else** structure or **decision** structure
- **Decision symbol:** a diamond shape in a flowchart that represents a decision/question



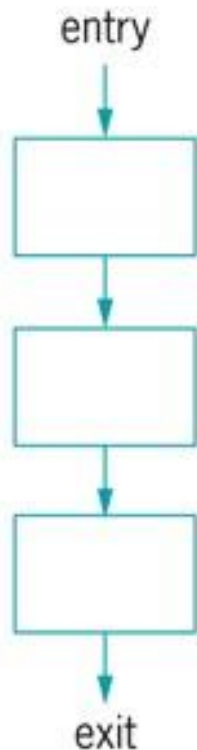
# The Repetition/Loop Structure

- Continues to repeat actions while a condition remains true
- Action or actions that occur within the loop are known as the **loop body**
- Programmers refer to looping as **repetition** or **iteration**
- **Infinite loop**: repeating flow of logic with no end

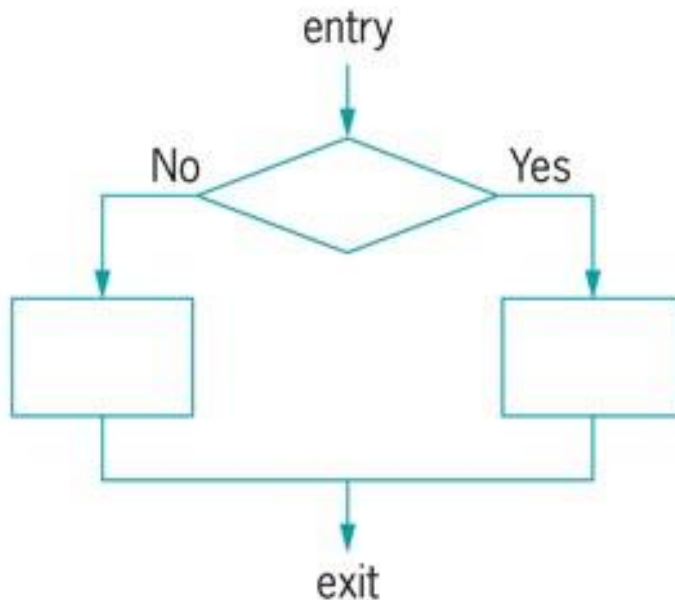


# Combining Structures

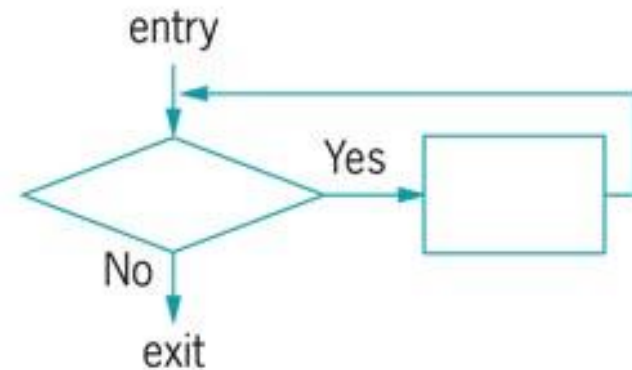
**Sequence**



**Selection**



**Loop**

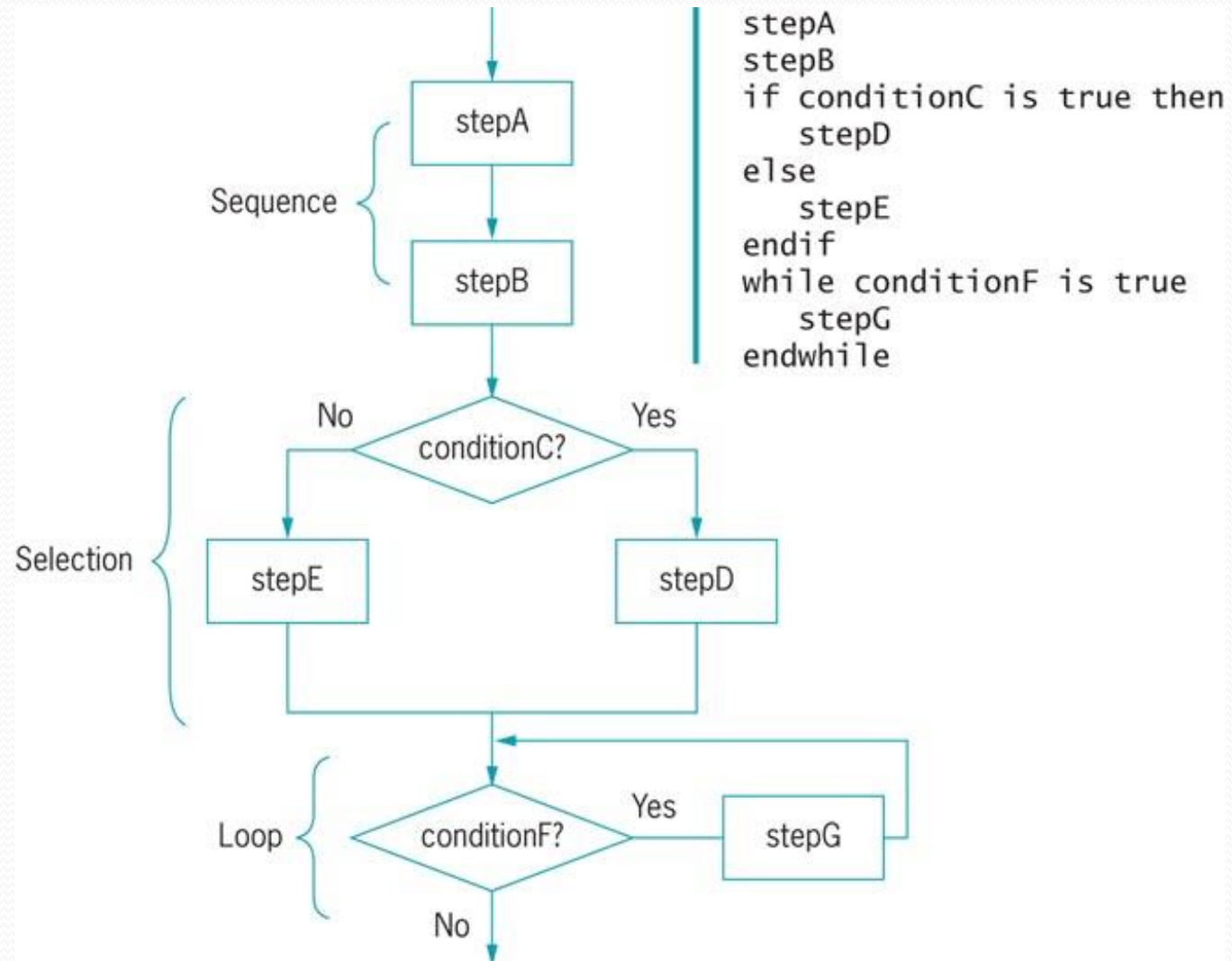


# Combining Structures

- A structured program includes only combinations of the three basic structures: sequence, selection, and loop
- Any structured program might contain one, two, or all three types of structures
- Structures can be stacked or connected to one another only at their entry or exit points
- Any structure can be nested within another structure

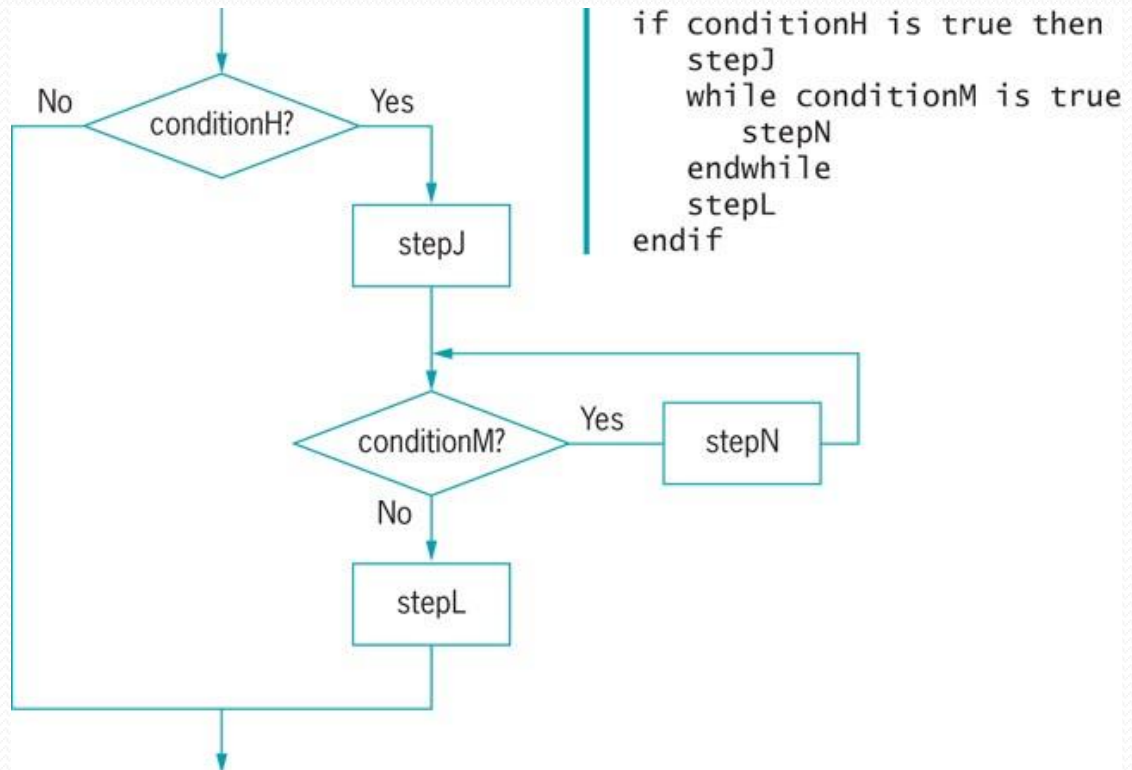
# Combining Structures

- Attaching structures end-to-end is called **stacking structures**



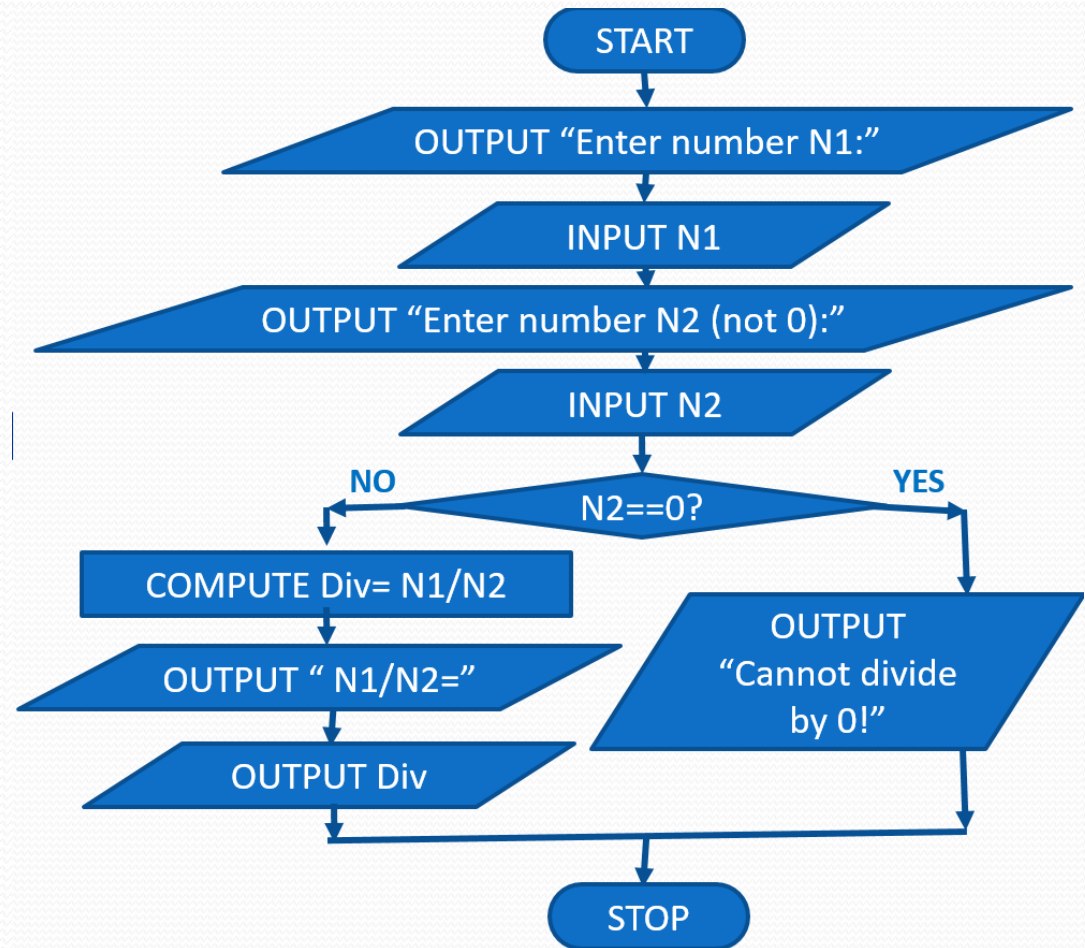
# Combining Structures

- Placing a structure within another structure is called **nesting structures**



# Division of 2 Numbers

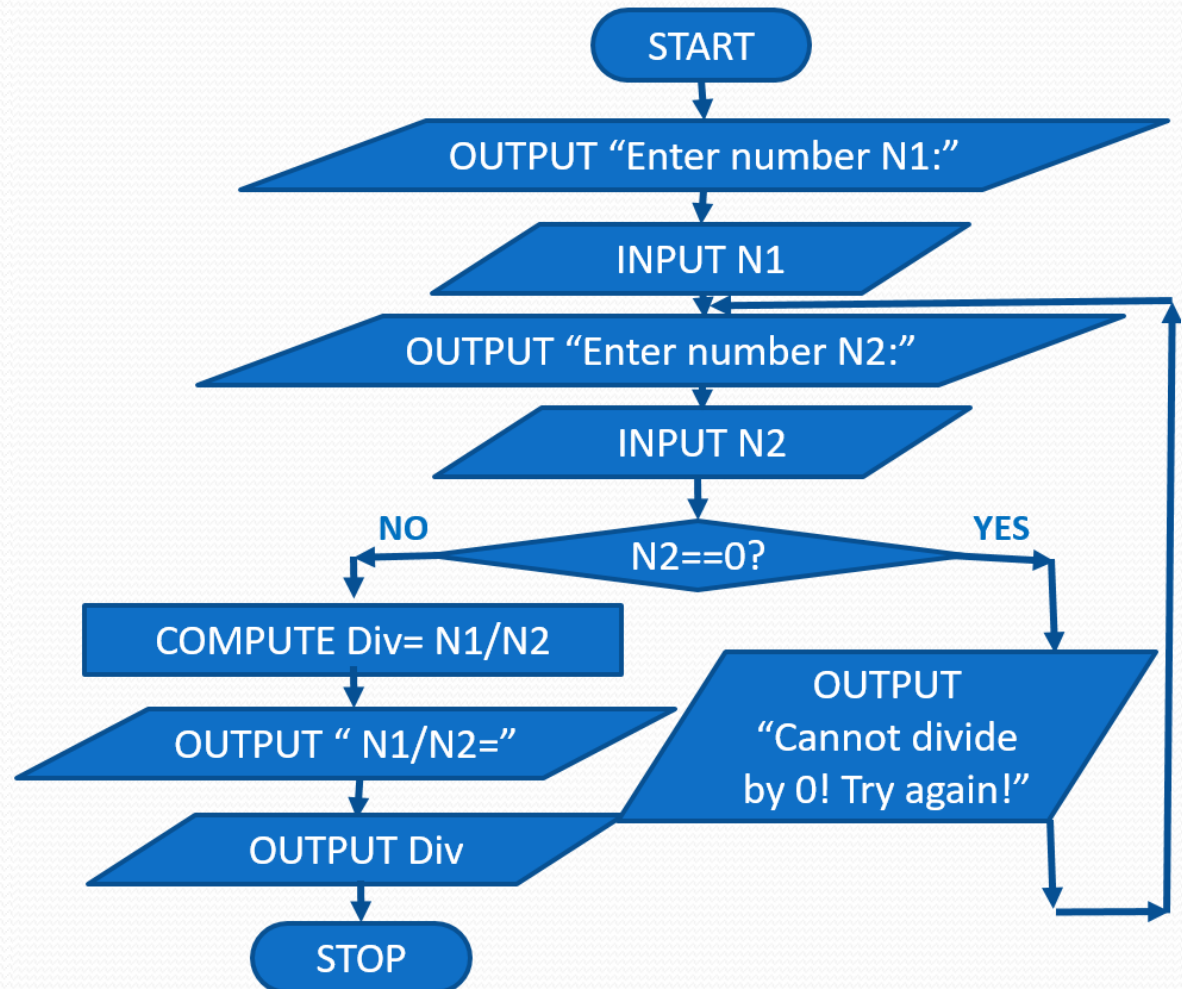
- Read 2 numbers N1 and N2 from the user and compute and output the division/quotient of the 2 numbers





# Division of 2 Numbers

- Read 2 numbers N1 and N2 from the user and compute and output the division/quotient of the 2 numbers



# Summary

- Programming Process and Problem-Solving Technique
- Flowcharts and Pseudocode
- Structures
- Examples and Exercises