

3

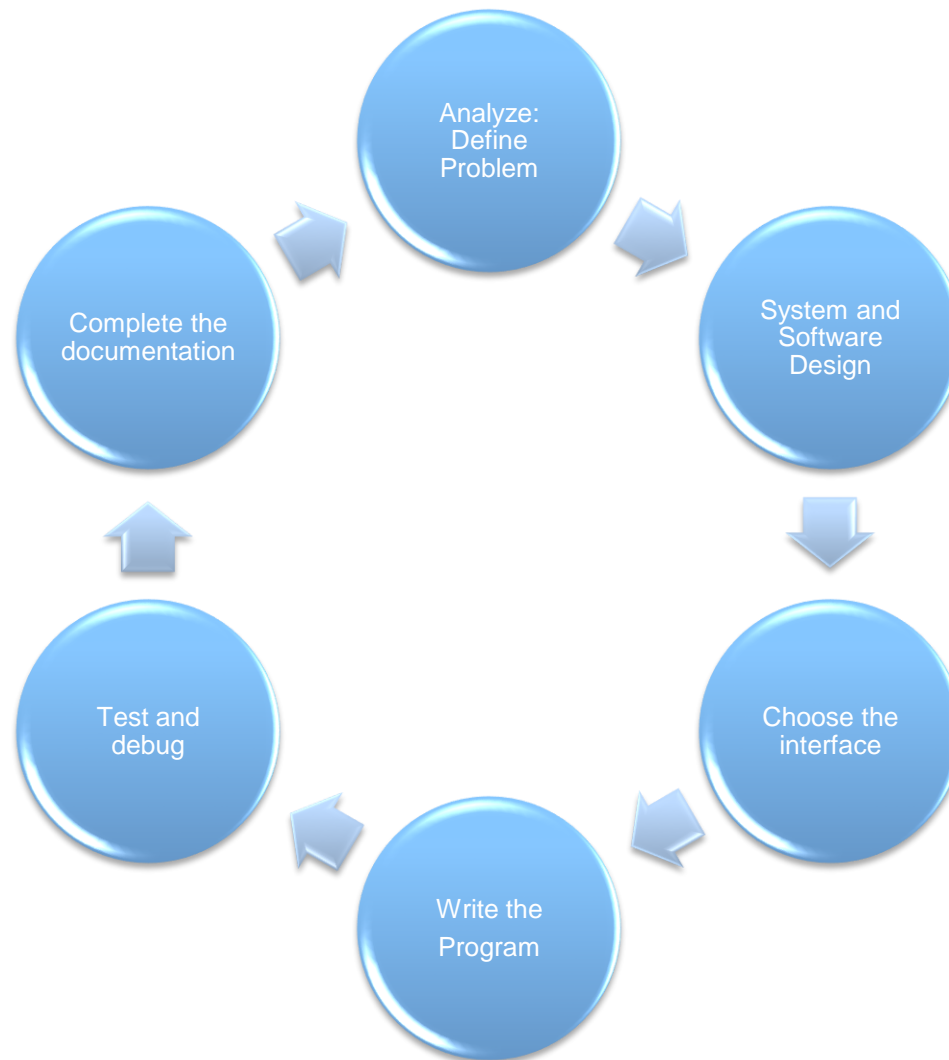
C How to Program

Program Design

Introduction

- Before writing a program:
 - Have a thorough understanding of the problem
 - Carefully plan an approach for solving it
- While writing a program:
 - Know what “building blocks” are available
 - Use good programming principles

Program Life cycle





3.1 Problem Solving

- Start with a **real-world problem** that needs to be **solved**
- Convert the real-world problem into a **computational** problem
- Develop an **algorithm** and use it to **solve** the computational problem

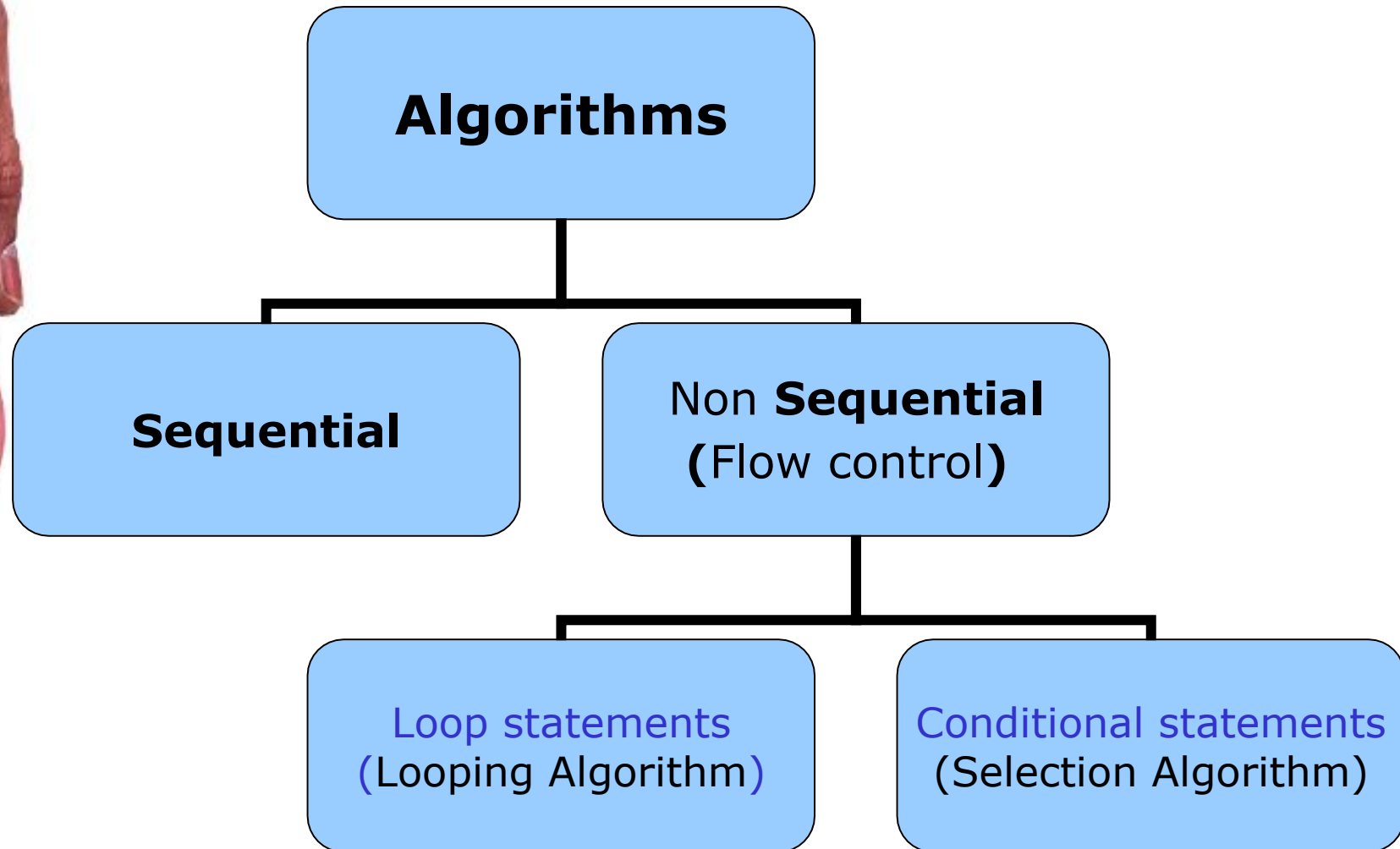
Algorithms

- Computing problems
 - All can be solved by executing a series of actions in a specific order
- Algorithm: procedure in terms of
 - Actions to be executed
 - The order in which these actions are to be executed
- Program control
 - Specify order in which statements are to be executed

Algorithm Example

- Consider the **problem** of computing the **sum of 2 numbers** which are entered by the user at the keyboard:
- Possible **algorithm** to solve the problem:
 - Read (i.e. input) the first number and store it in the computer
 - Read the second number and store it in the computer
 - Add the first number to second number to produce a sum, SUM. Store SUM in the computer
 - Display (i.e. output) the value of SUM

Algorithms' Kinds



3.2 Design Tool: Pseudocode

- Pseudocode
 - Artificial, generic way of describing an algorithm, without use of any specific programming language
 - Helps programmers to plan an algorithm
 - Similar to everyday English
 - Not an **actual programming language**, but may borrow syntax from popular programming languages
 - Not actually executed on computers
 - Styles vary
 - Helps us “think out” a program before writing it
 - Easy to convert into a corresponding C program
 - Consists only of executable statements

Pseudocode Example

- **Example Problem:** Calculate the bill when someone buys a specific number of some item:
- **Pseudocode:**
 - PROMPT for number of items being purchased
 - READ number of items being purchased
 - PROMPT for price per item
 - READ price per item
 - CALCULATE subtotal
 - CALCULATE tax
 - CALCULATE total
 - DISPLAY total

Common Pseudocode Keywords

- **When describing input, output, computations, etc, the following terms are often used:**
 - Input: INPUT, READ, GET
 - Output: PRINT, DISPLAY, SHOW, PROMPT
 - Compute: COMPUTE, CALCULATE, DETERMINE
 - Initialize: SET, INIT
 - Add one: INCREMENT, BUMP
 - Decisions: TEST, IF/ELSE, WHILE

Writing Pseudocode Algorithms

- Be certain the task is completely specified!
- Questions to ask:
 - What data is **known** before the program runs?
 - What data must be **input** by the user?
 - What **computations** will be performed on the data?
 - What data will be **output** (displayed) to the user?

3.2.1 Pseudocode Sequential Algorithm Example (1/2)

- **Real World Problem:**
 - Calculate your two children's' allowances, based upon 75 cents per year old.
- **Known Values**
 - Rate = 75 cents per year
- **Inputs**
 - Ages of children
- **Calculations**
 - Allowance = Age x Rate
- **Outputs**
 - Allowances for each child

Pseudocode Sequential Algorithm Example (2/2)

- **Pseudocode Algorithm:**

- PROMPT for Age of Child1
- READ Age of Child1
- PROMPT for Age of Child2
- READ Age of Child2
- CALCULATE

$\text{Allowance for Child1} = \text{Age of Child1} \times \text{Rate}$

- CALCULATE

$\text{Allowance for Child2} = \text{Age of Child2} \times \text{Rate}$

- DISPLAY Allowance for Child1
- DISPLAY Allowance for Child2

Sequential Program Statements

- With **sequential** program statements, you execute **one statement after another** (like steps in a list)
 - After step X is completed, step X+1 will be performed, until the last statement has been executed.
- **Every** step in the list **will** be performed.
- **Each** step in the list will be performed **once and only once**.

Non-Sequential Program Statements

- Programs that execute only sequential program statements are pretty simplistic.
- Most programs need more **flexibility** in the order of statement execution.
- The order of statement execution is called the **flow of control**.
- **Control** statements allow the execution of statements based upon **decisions**.

3.2.2 Flow of Control Statements

- Conditional statements:
 - Decide **whether or not** to execute a particular statement or statements
 - Also called the **selection** statements or **decision** statements.
- Pseudocode Example:
IF HoursWorked over 40
 DISPLAY overtime pay
ELSE DISPLAY regular pay

3.2.2.1 Pseudocode Selection Algorithm Example (1/2)

- **Real World Problem:**
 - Calculate one child's allowance, based upon 75 cents per year old if s/he is under 10, and \$1 per year if 10 or over.
- **Known Values**
 - YoungRate = 75 cents per year
 - OlderRate = 100 cents per year
 - BreakAge = 10
- **Inputs**
 - Age of child
- **Calculations**
 - Allowance = Age x Rate
- **Outputs**
 - Allowance

Pseudocode Selection Algorithm Example (2/2)

- **Pseudocode Algorithm:**
 - PROMPT for Age
 - READ Age
 - IF Age less than 10
 - THEN CALCULATE Allowance = Age x YoungRate
 - ELSE CALCULATE Allowance = Age x OlderRate
 - DISPLAY Allowance

Flow of Control Statements

- Loop statements:
 - Repeatedly execute the same statement(s) for a certain number of times **or** until a test condition is satisfied.
- Pseudocode Example:
WHILE Population UNDER Limit
 COMPUTE Population = Population + Births - Deaths





3.2.2.2 Pseudocode Looping Algorithm Example (1/2)

- **Real World Problem:**
 - Calculate the total allowance paid to each of three children at \$1 per year old.
- **Known Values**
 - Rate = \$1 per year
 - NumKids = 3
- **Inputs**
 - Ages of children
- **Calculations**
 - Allowance = Age x Rate
- **Outputs**
 - Total Allowance paid

Pseudocode Looping Algorithm Example (2/2)

- **Pseudocode Algorithm:**
 - Set KidsPaid to 0
 - Set Total to 0
 - WHILE KidsPaid under 3
 - PROMPT for Age
 - Read Age
 - CALCULATE Allowance = Age x Rate
 - ADD Allowance to Total
 - INCREMENT KidsPaid
 - DISPLAY Total

3.3 Flow Control Design Tool: Flowcharts

- Flowchart - a graphical way of writing algorithms
 - Rectangle is used for calculations 
 - Parallelogram is used for input and output 
 - Circle is used as connector 
 - Diamond is used as decision 
 - Symbols are connected by arrows to represent the order of the operations

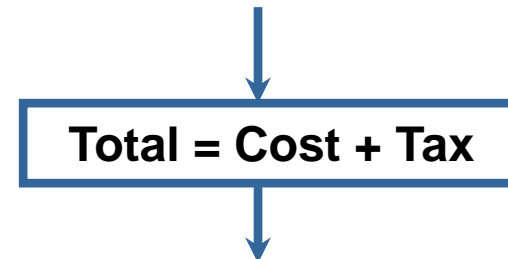


Flowcharts Symbols: Calculations

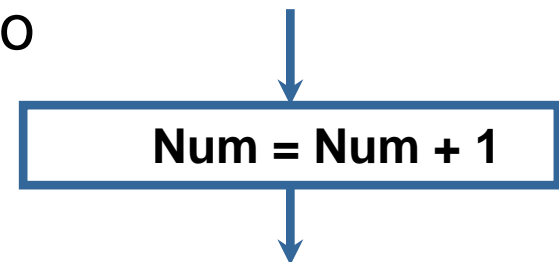
- **Calculations** (e.g. arithmetic expressions) are shown in **rectangles**

- Examples:

- **Total = Cost + Tax**

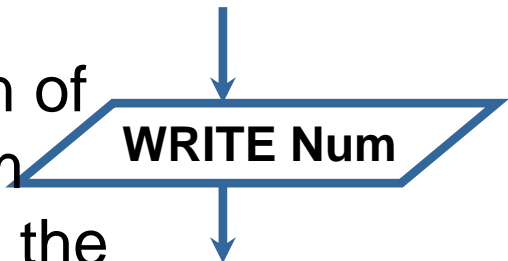
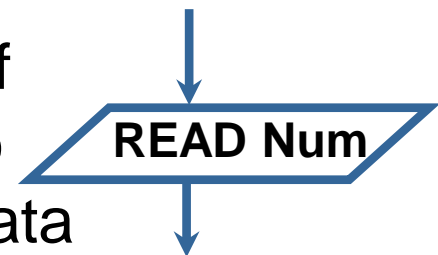


- **Num = Num + 1** (add one to the current value of Num and make that the new value of Num)



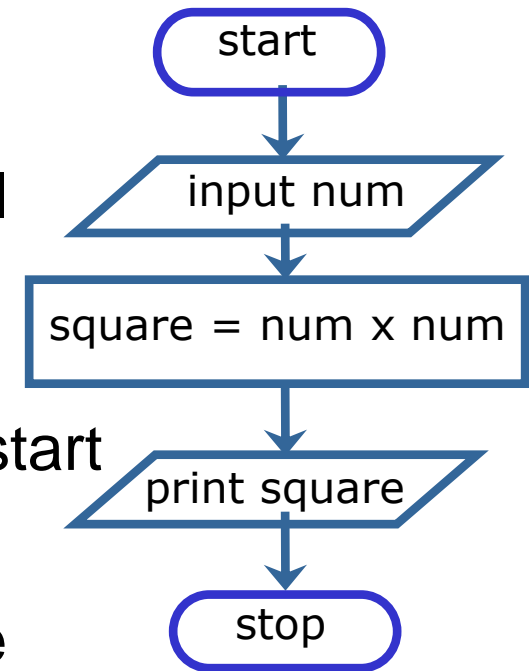
Flowcharts Symbols: Input/Output

- Data **input** and **output** are shown in **parallelograms**
- **Input** means a **read** operation of data from a peripheral device to memory (e.g. a user typing in data at a keyboard)
- **Output** means a **write** operation of data to a peripheral device from memory (e.g. data displayed to the monitor)



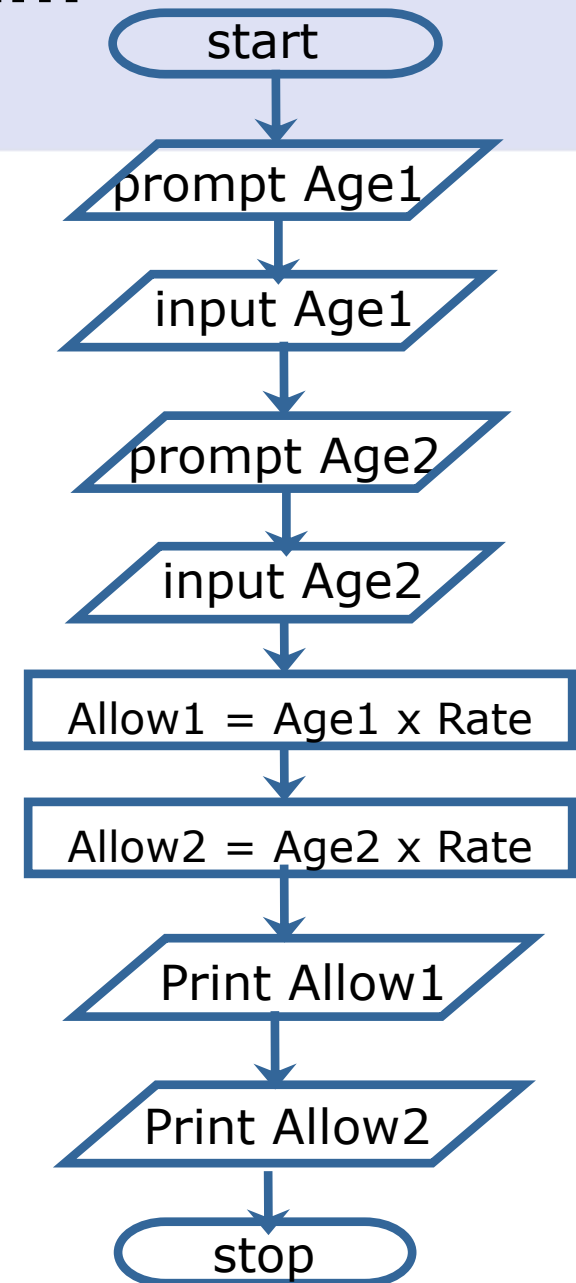
Flowcharts Symbols: Start and End

- Every algorithm starts somewhere and terminates somewhere
- Every flowchart must have **one start** symbol and **one end** symbol
- Start and end symbols are **ovals**
 - A start symbol denotes the start of the algorithm
 - An end symbol indicates the algorithm has terminated

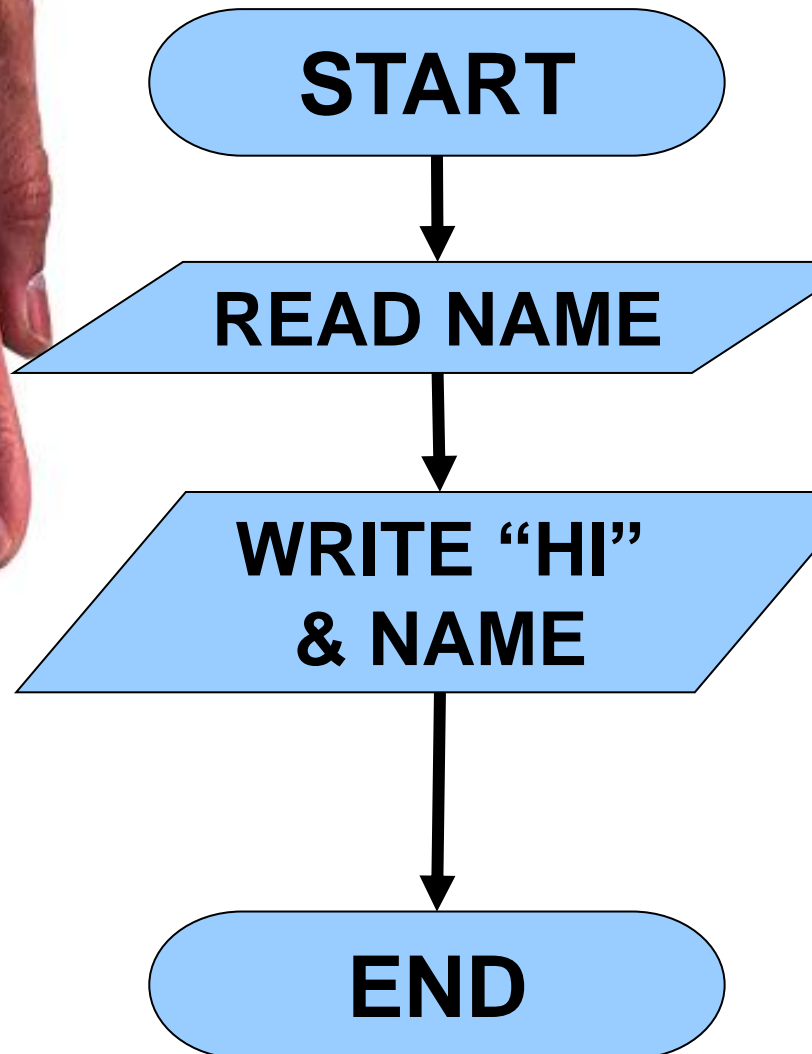


3.3.1 Sequential Algorithm Flowchart

- **Previous Pseudocode Algorithm:**
 - PROMPT for Age of Child1
 - READ Age of Child1
 - PROMPT for Age of Child2
 - READ Age of Child2
 - CALCULATE Allowance for Child1 = Age of Child1 x Rate
 - CALCULATE Allowance for Child2 = Age of Child2 x Rate
 - DISPLAY Allowance for Child1
 - DISPLAY Allowance for Child2



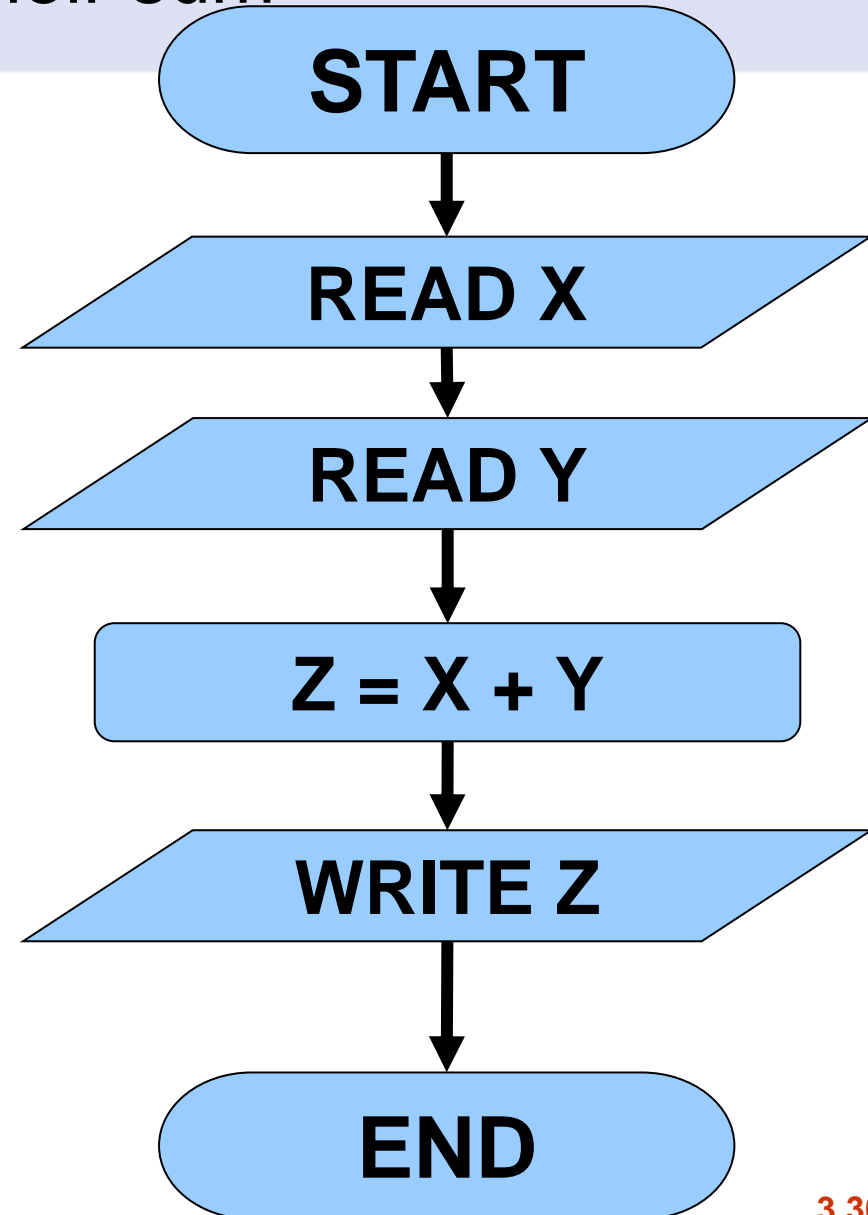
Example: a program is required to read my name and give me a “Hi My name “ message



We should have
This is
Sequence control
structure

Hi Name
Write the answer on
the screen

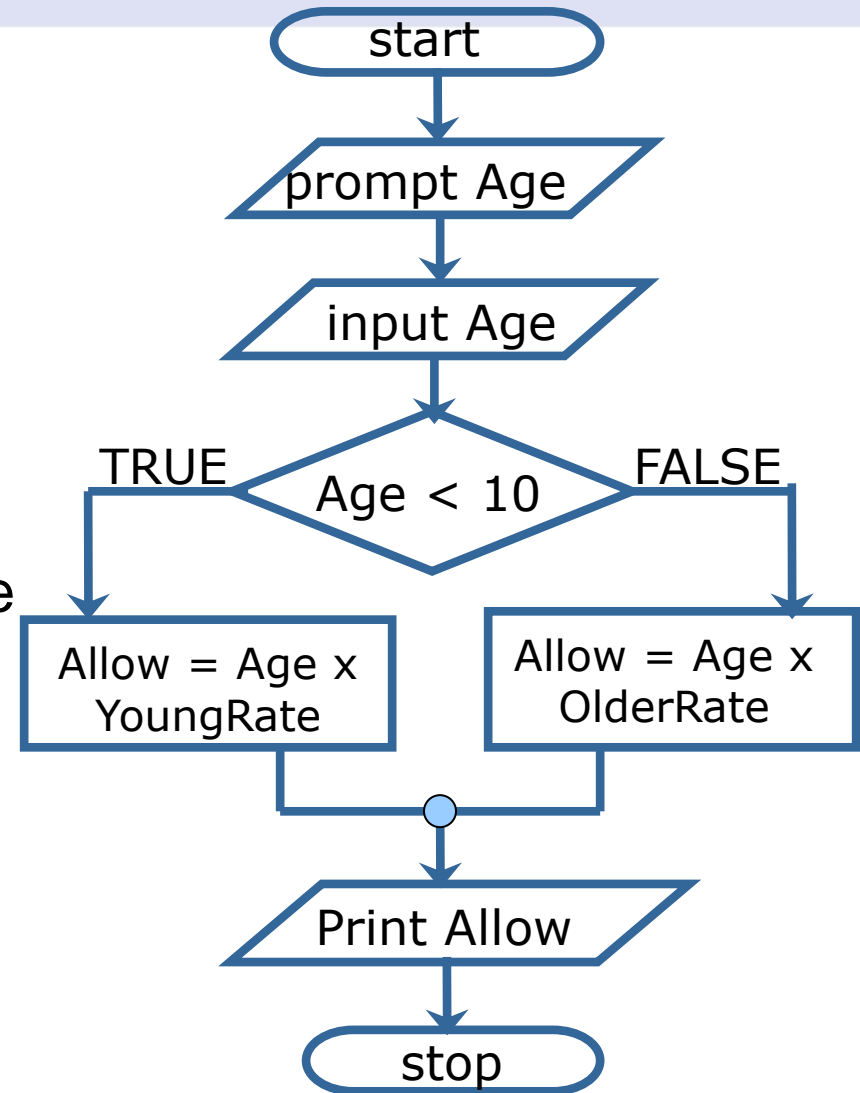
Example: a program is required to read two numbers and show their sum



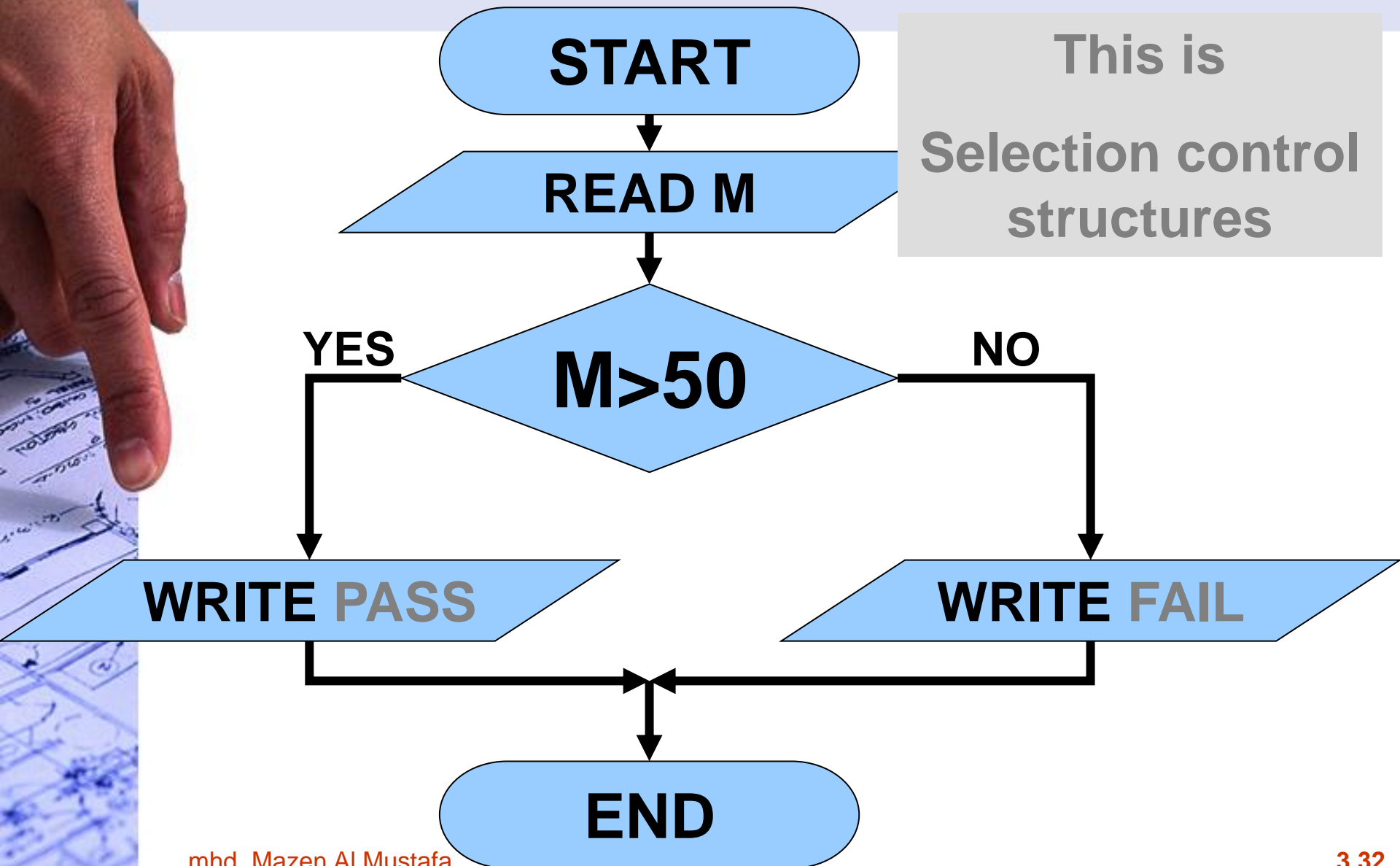
3.3.1.1 Selection Algorithm Flowchart

- **Previous Pseudocode Algorithm:**

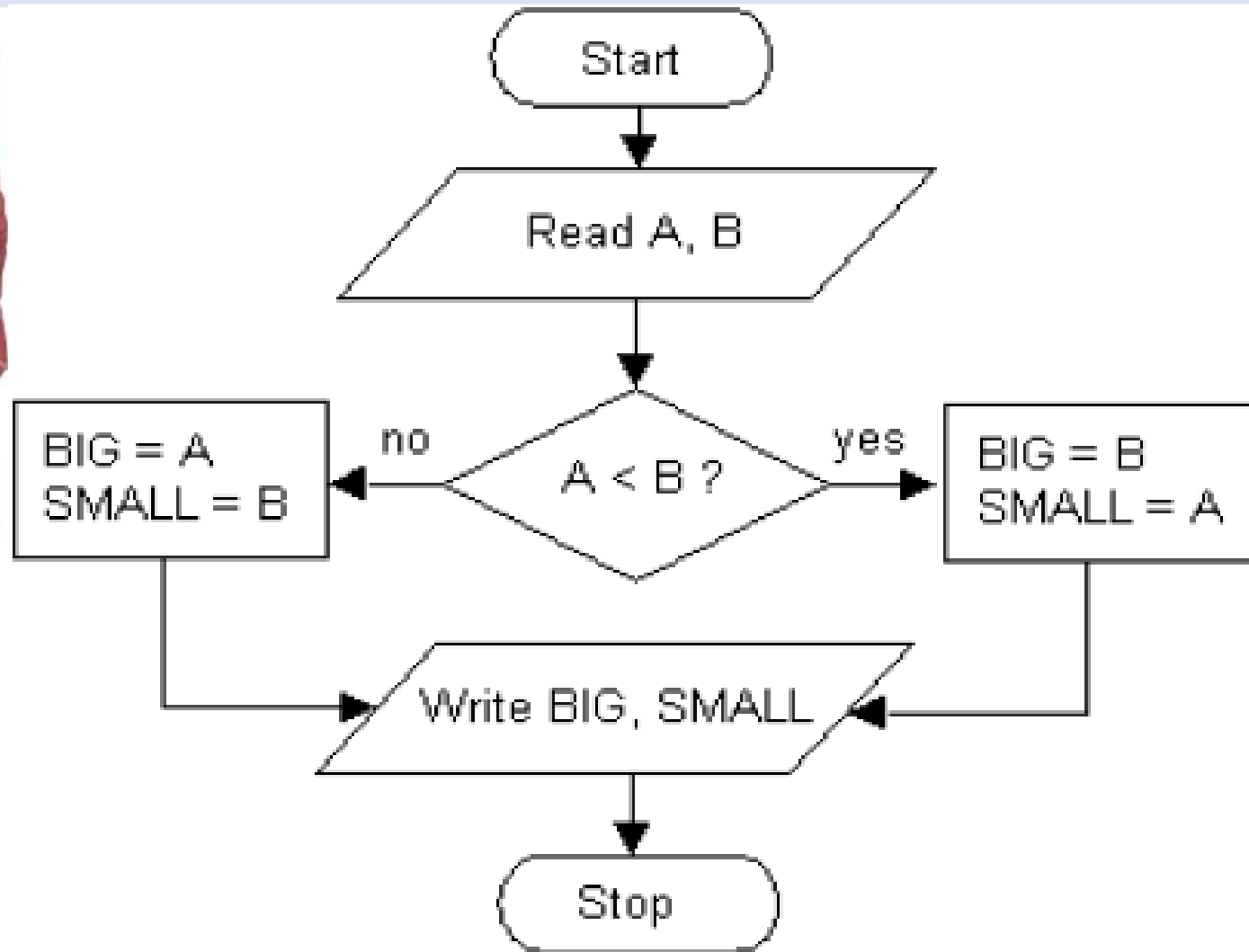
- PROMPT for Age
- READ Age
- IF Age less than 10
 - CALCULATE Allowance = Age x YoungRate
 - ELSE CALCULATE Allowance = Age x OlderRate
- DISPLAY Allowance



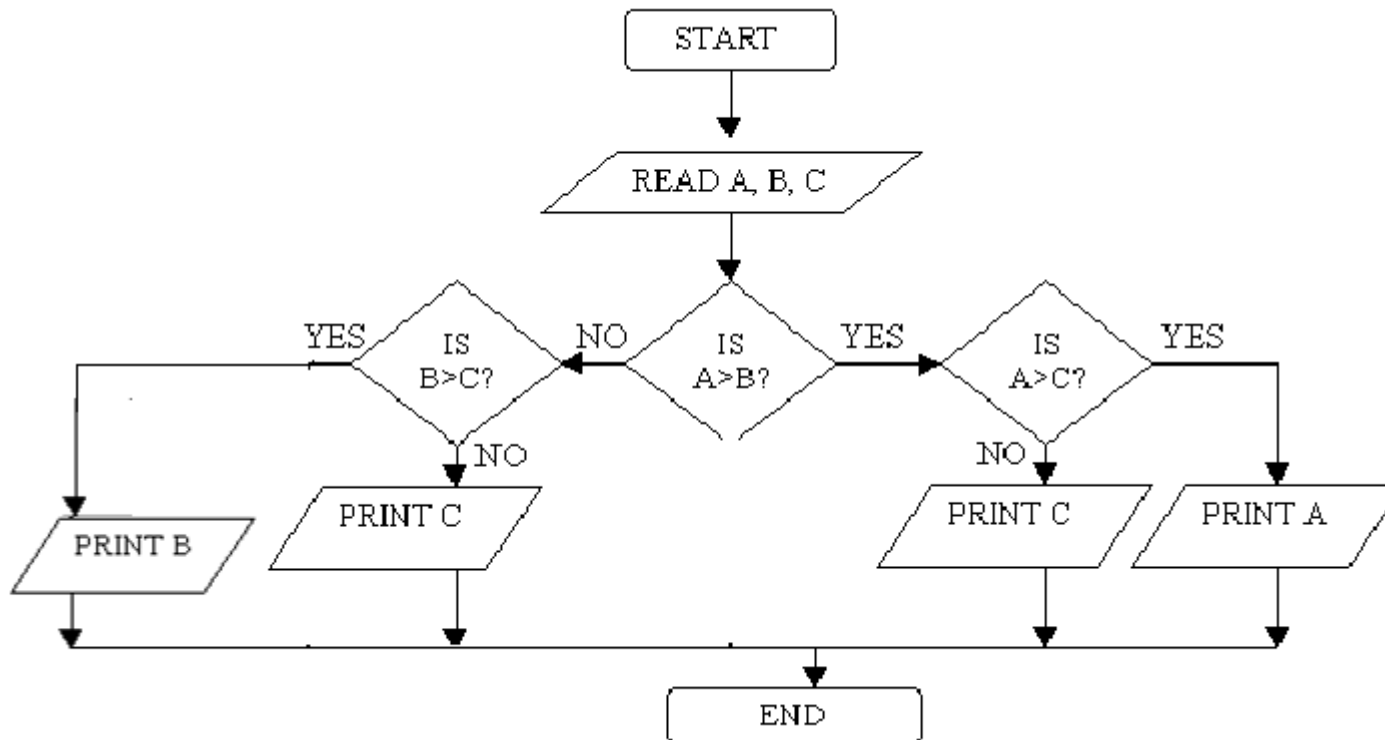
Example: a program read student mark and show result **PASS** if over 50, and **FAIL** if not.



Example: read two numbers find the BIG and the SMALL one



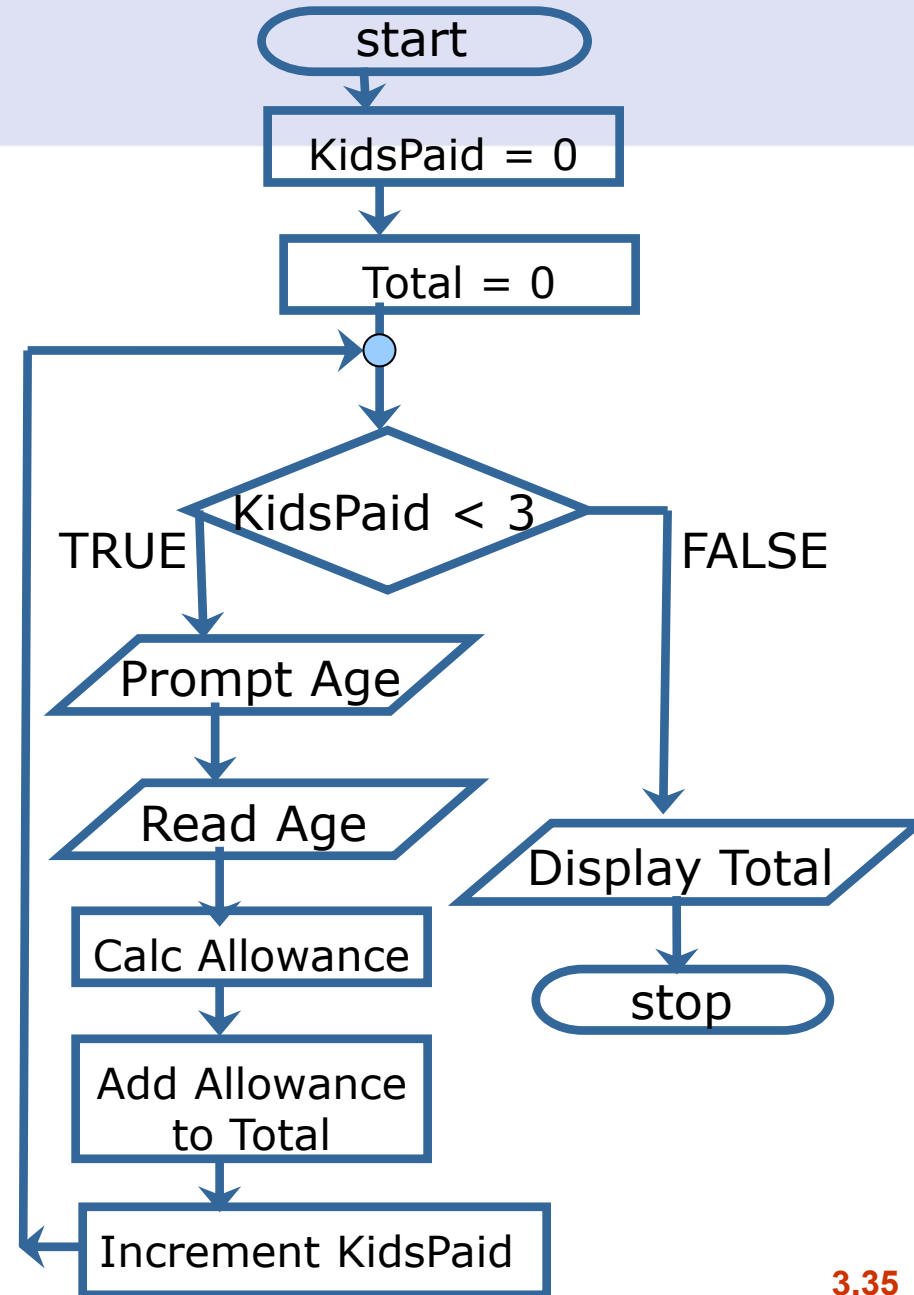
Example Draw a flowchart to find the largest of three numbers A, B, and C.



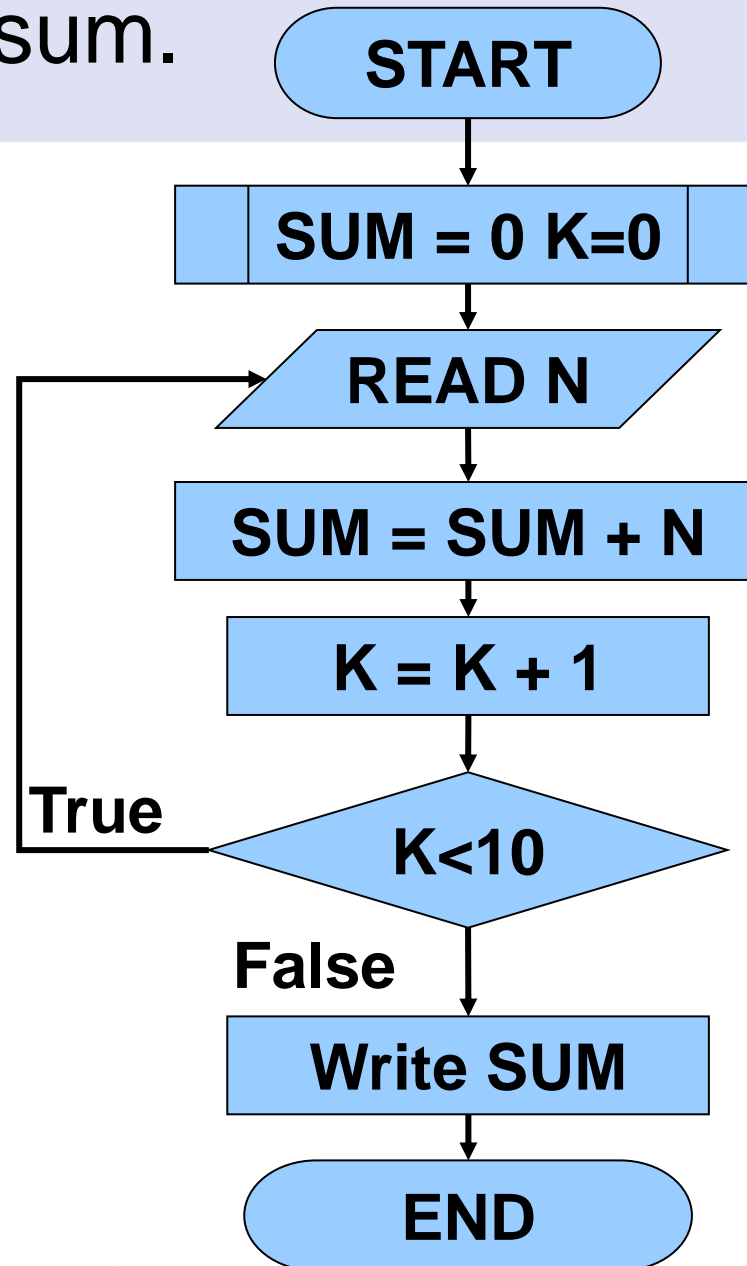
3.3.1.2 Looping Algorithm Flowchart

- **Previous Pseudocode Algorithm:**

- Set KidsPaid to 0
- Set Total to 0
- WHILE KidsPaid < 3
 - PROMPT for Age
 - READ Age
 - CALCULATE
Allowance = Age x Rate
 - ADD Allowance to Total
 - INCREMENT KidsPaid
- DISPLAY Total

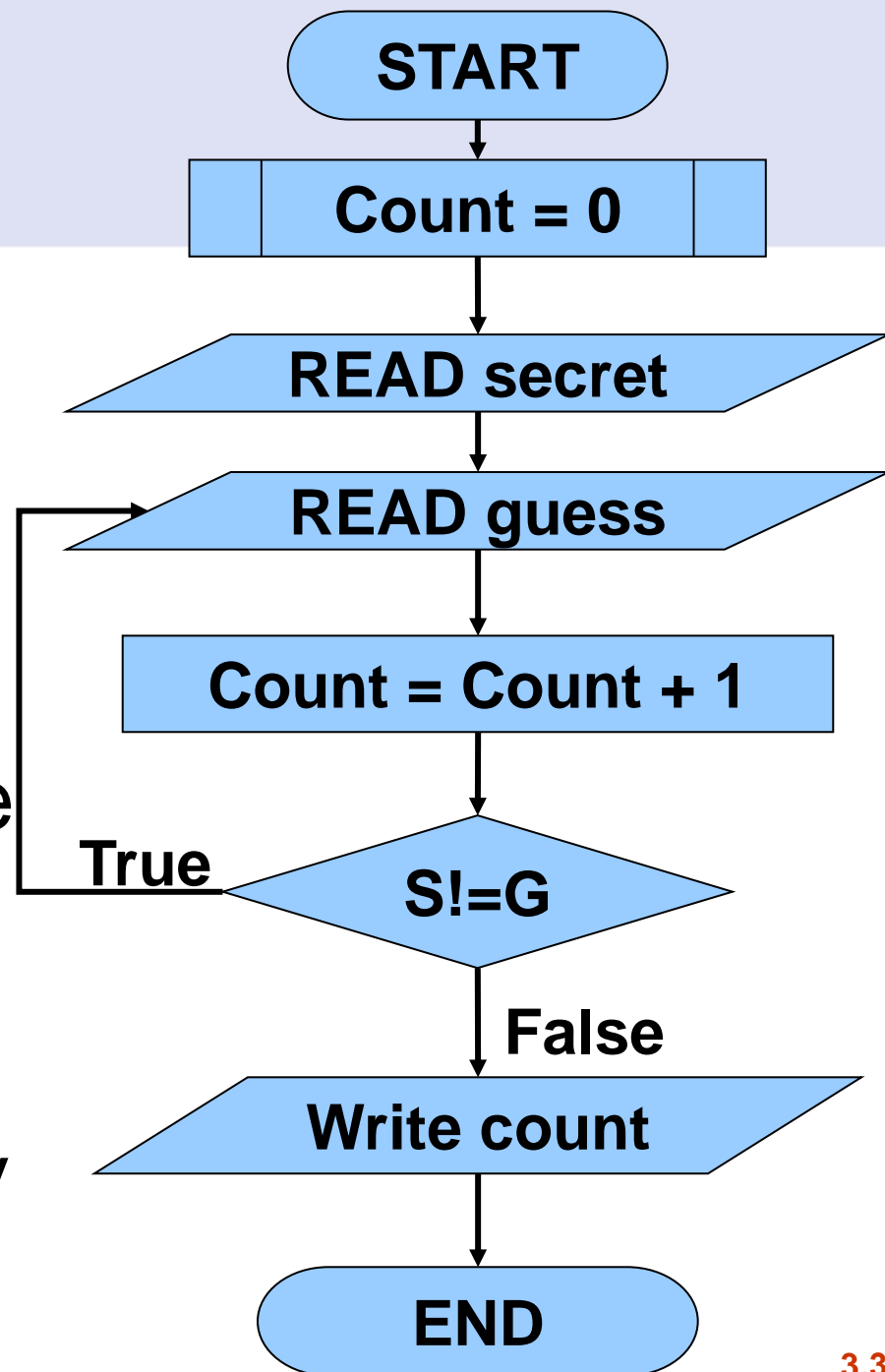


Example: read 10 numbers and find their sum.



Example:

1. Player1 enter **SECRET** number
2. Player 2 have to guess it. He enter **GUESS** numbers until he find the secret number.
3. Show how many times he try



Assignment 1

- Write a Pseudocode and draw a flowchart to the next problem

A hand is visible on the left side of the image, pointing with the index finger towards a document. The document appears to be a technical drawing or blueprint, with various lines and text visible. The background of the slide is a light blue gradient.

Someone Stole a Cookie from the Cookie Jar

Problem: Mamma had just filled the cookie jar when the 3 children went to bed. That night one child woke up, ate half of the cookies and went back to bed. Later, the second child woke up, ate half of the remaining cookies, and went back to bed. Still later, the third child woke up, ate half of the remaining cookies, leaving 3 cookies in the jar. How many cookies were in the jar to begin with?

Specific Solution to the Problem

- First, we solve the specific problem to help us identify the steps.
 - 3 cookies left $\times 2 = 6$ cookies left after 2nd child
 - $6 \times 2 = 12$ cookies left after 1st child
 - $12 \times 2 = 24 =$ original number of cookies

A Generic Algorithm

- What is a **generic algorithm** for this problem?

An algorithm that will work with any number of remaining cookies

AND

that will work with any number of children.

Generic Algorithm for Cookie Problem

- Get number of children.
- Get number of cookies remaining.
- While there are still children that have not raided the cookie jar, multiply the number of cookies by 2 and reduce the number of children by 1.
- Display the original number of cookies.

Assignment 2

- Write a Pseudocode and draw a flowchart to the next problem

Ahmad's Shopping Trip

Problem: Brian bought a belt for \$9 and a T-shirt that cost 4 times as much as the belt. He then had \$10. How much money did Brian have before he bought the belt and T-shirt?

Specific Solution

$$\text{Start\$} = \text{Belt\$} + \text{Shirt\$} + \$10$$

$$\text{Start\$} = \text{Belt\$} + (4 \times \text{Belt\$}) + \$10$$

$$\text{Start\$} = 9 + (4 \times 9) + 10 = \$55$$

Generic Algorithm

- Now, let's write a generic algorithm to solve any problem of this type.
- What are the inputs to the algorithm?
 - the **cost** of the first item (doesn't matter that it's a belt): **item1 price**
 - the **number** to multiply the cost of the first item by to get the cost of the second item: **multiplier**
 - the **amount** of money left at the end of shopping: **amount left**