Week# 2
Problem solving techniques

Dr Taimur Ahmed

Department of IT & CS, PAF-IAST

Algorithms

- What is an **Algorithm**?
  - *Algorithm is a **step-by-step procedure**, which defines a set of instructions to be executed in a **certain order** to get the desired output*
- Algorithms are **general** and are created **independent of any language**
- An algorithm can be implemented for **more than one** programming language
- An Algorithm is also referred as "***Pseudocode***"
  - *Pseudo* means **false** – Pseudocode means a **false program**

Algorithms – Advantages & Disadvantages

❑ **Advantages** of Algorithms

- Well defined algorithms are **easy to understand** the problem
- Algorithm is a **step-wise representation** of a solution to a given problem
- In an Algorithm the problem is broken down into **smaller pieces or steps** hence, it is easier for the programmer to convert it into an actual program

❑ **Disadvantages** of Algorithms

- Writing an algorithm **takes a long time** so it is time-consuming
- Branching and Looping statements are difficult to show in Algorithms

Characteristics of an Algorithm

- **Unambiguous** – Algorithm should be **clear and unambiguous**. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning
- **Input** – An algorithm should have **0 or more** well-defined inputs.
- **Output** – An algorithm should have **1 or more** well-defined outputs, and should match the desired output
- **Finite-ness** – Algorithms **must terminate** after a finite number of steps
- **Feasibility** – Should be **feasible** with the available resources
- **Independent** – An algorithm should have step-by-step directions, which should be **independent of any programming code**

How to Design and Write an Algorithm?

- There are **no well-defined standards** for writing algorithms
  - Rather, they are problem and resource dependent
- Algorithms are **never** written to support a particular programming code
- To write an Algorithm, following things are needed
  1. A **problem** that is to be solved by this algorithm
  2. The **constraints** of the problem that must be considered while solving the problem
  3. The **input** to be taken to solve the problem
  4. The **output** to be expected when the problem the is solved
  5. The **solution** to this problem, in the given constraints

Writing an Algorithm - Examples

❑ Example 1

Problem – Design an algorithm to **add two numbers** and **display the result**

Step 1: START
Step 2: declare three integers **a, b & c**
Step 3: define values of **a & b**
Step 4: add values of **a & b**
Step 5: store output of **Step 4** in **c**
Step 6: print **c**
Step 7: STOP

Writing an Algorithm - Examples

❑ Example 1 – Alternative method

Problem – Design an algorithm to **add two numbers** and **display the result**

Step 1: START
Step 2: get values of **a & b**
Step 3: **c** ← **a & b**
Step 4: display **c**
Step 5: STOP

Flowcharts in Programming

Flowcharts in Programming

- What is a **flowchart**?
  - Flowchart **is a diagram/visual representation** of an algorithm
  - A flowchart can be helpful for both **writing programs and explaining programs** to others
- A flowchart uses **symbols** which are connected with each other to indicate the **flow of information and processing**
- The process of drawing a flowchart for an Algorithm is known as "**flowcharting**"

Basic Symbols in Flowcharts

Symbol	Purpose	Description
	Flow line	Indicates the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Represents the start and the end of a flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.

Basic Symbols in Flowcharts

	Decision	Used for decision making between two or more alternatives.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect the flowchart portion on a different page.
	Predefined Process/Function	Represents a group of statements performing one processing task.

Flowchart – Example

Problem – <u>Add two numbers</u> entered by the user

Algorithm

Step 1: START

Step 2: declare three integers **n1, n2** & **sum**

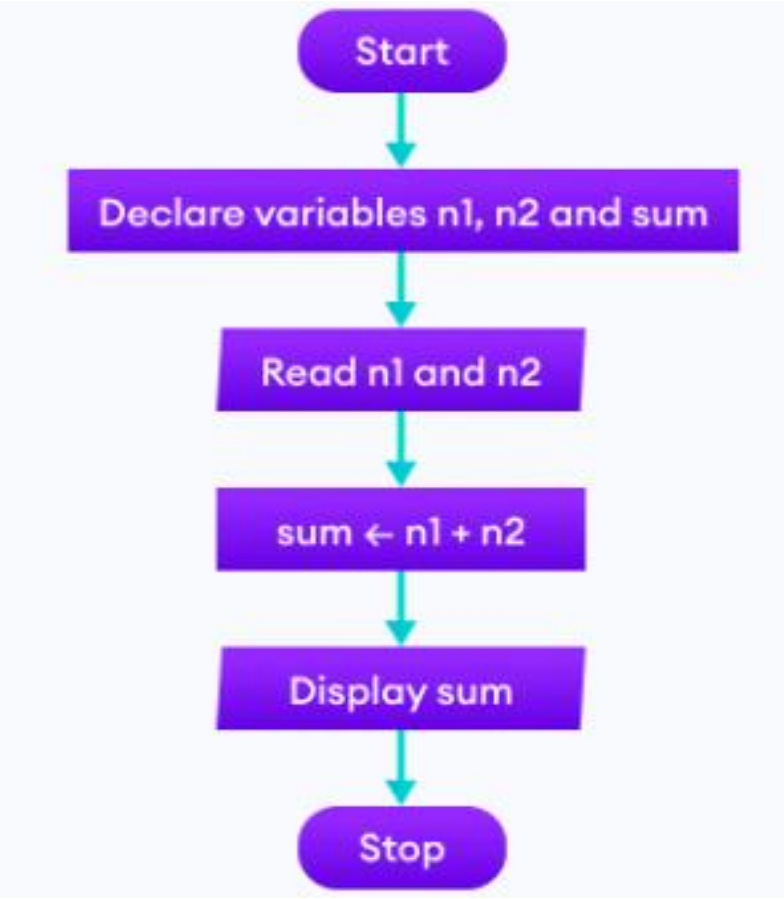
Step 3: define values of **n1** & **n2**

Step 4: add values of **n1** & **n2**

Step 5: print **sum**

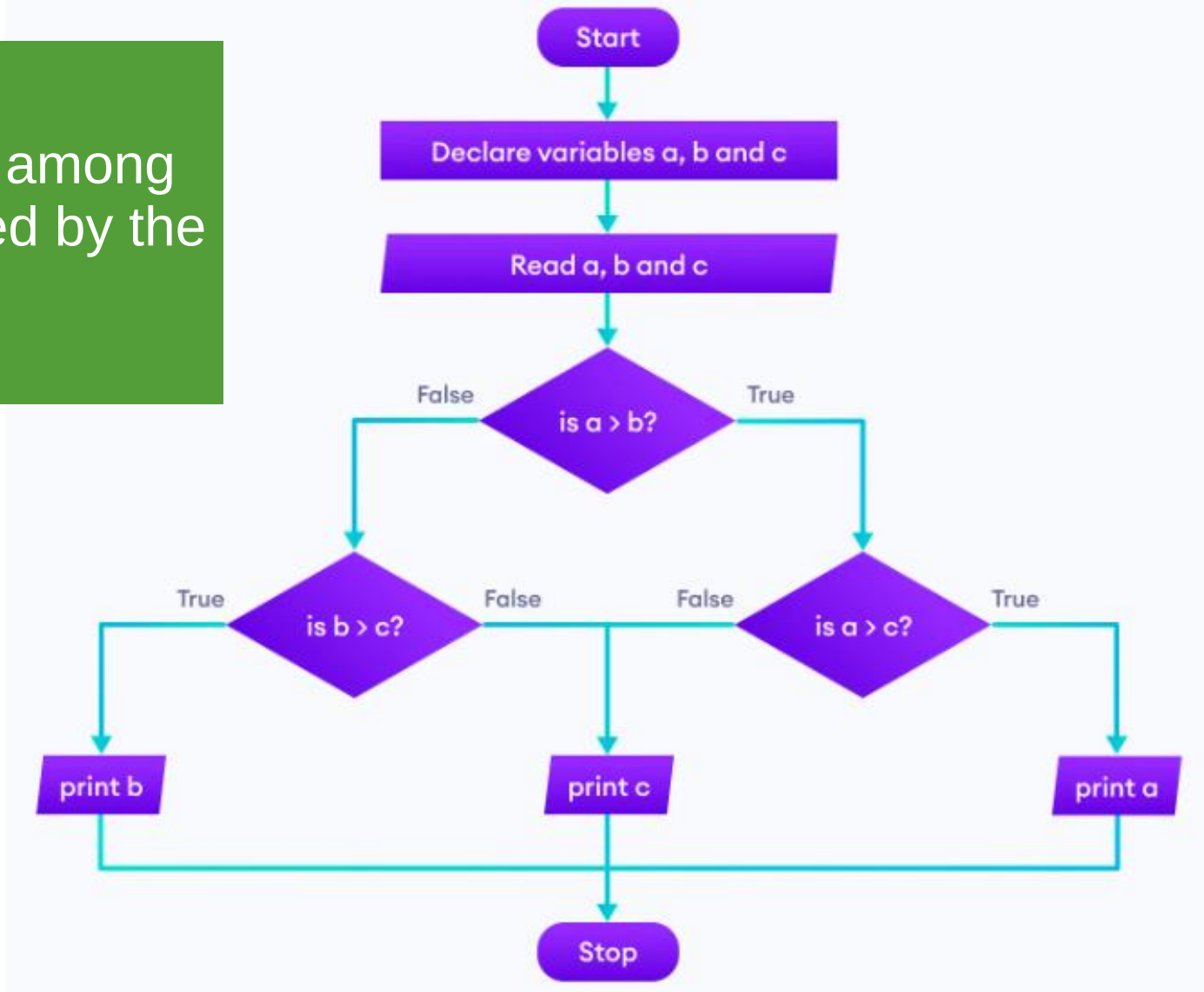
Step 6: STOP

Flowchart



Flowchart – Example

Problem – Find the **largest** among **three different numbers** entered by the user



Flowcharts – Advantages & Disadvantages

❑ Advantages of Flowcharts

- Flowcharts are better way of communicating the **logic of a problem**
- Flowcharts act as a guide for **blueprint** during program designed
- Flowcharts help in **debugging process**
- With the help of flowcharts programs can be **easily analyzed**
- Flowcharts serve as a **good proper documentation**

❑ Disadvantages of Flowcharts

- It is difficult to draw flowchart for **large and complex programs**
- There is **no standard** to determine the amount of detail
- Difficult to **reproduce** the flowcharts
- It is difficult to **modify the Flowchart**

Practice problems!

- Write **Algorithms** and **draw flow charts** for the following problems;
  1. Take value of today’s temperature from user in degree centigrade and display converted temperature in Kalvin.
  2. Take height of user in foot and inches and display user’s height in centimetres.
  3. Take number of years from user and print total number of days in the input number of years.