

L#3. Problem Solving Tools and Program Design in Computer Programming: Sequential Problems

PART 1

Problems with solutions requiring sequential process

Version: Jan 2020 (31 slides)

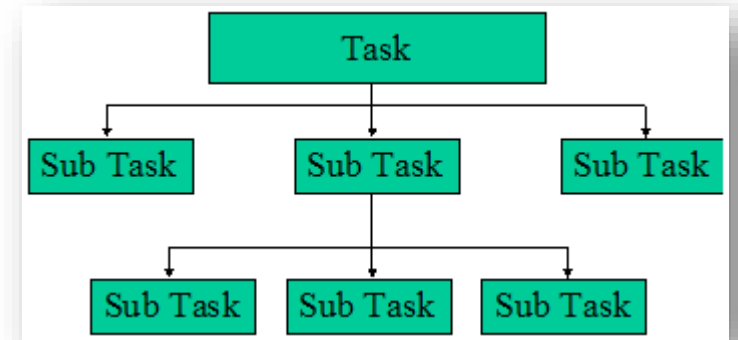
DR. mARCO A ARΘhA

Steps in Program Development

1. *State the Problem Clearly*- a problem cannot be solved correctly unless it is being understood. Identify Input/Output (e.g., decompose it with IPO)
2. *Design the Program*. Write an algorithm. Using plain words, write the logical order of instructions. The computer follows the directions exactly at the given sequence. Use top down design. Write the algorithm using pseudocode or flowchart.
3. *Code the Program* - Write the programming statements in the desired language (e.g., Python, C, Fortran, matlab).
4. *Run, Test, and Debug the program* - Check if you get the desired output, if not, trace the possible error(s) and fix them.

Program Design: Top Down Design

- Program design is the development of an algorithm to solve a problem.
- An ALGORITHM is merely *the sequence of steps* taken to solve a problem. It is a *recipe*. The steps involve programming structures: 'sequence', 'selection' and 'loops'
- We will use a method called TOP DOWN DESIGN, which is the best defense against confusion and chaos. It's a disciplined model for designing programs allowing even the largest programs to be broken into manageable pieces.
- By definition, an algorithm is independent of any particular programming language, so program design must avoid relying on the features of a given computer language.

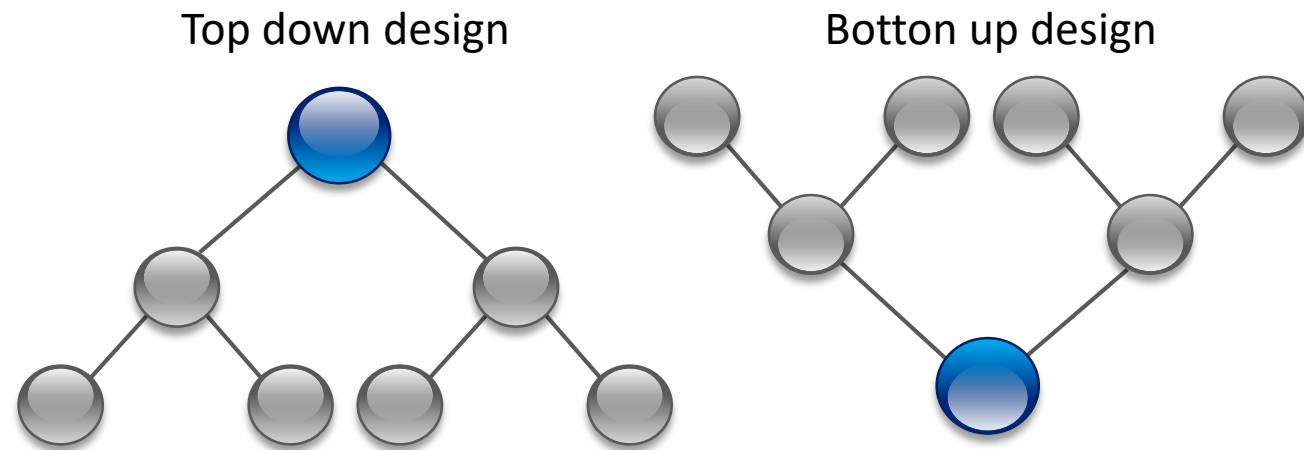


Top Down Design is also called Modular Programming

Program Design: Top Down Design

- Program design is the development of an algorithm to solve a problem.
- An ALGORITHM is merely *the sequence of steps* taken to solve a problem. It is a *recipe*. The steps involve programming structures: 'sequence', 'selection' and 'loops'
- We will use a method called TOP DOWN DESIGN, which is the best defense against confusion and chaos. It's a disciplined model for designing programs allowing even the largest programs to be broken into manageable pieces.

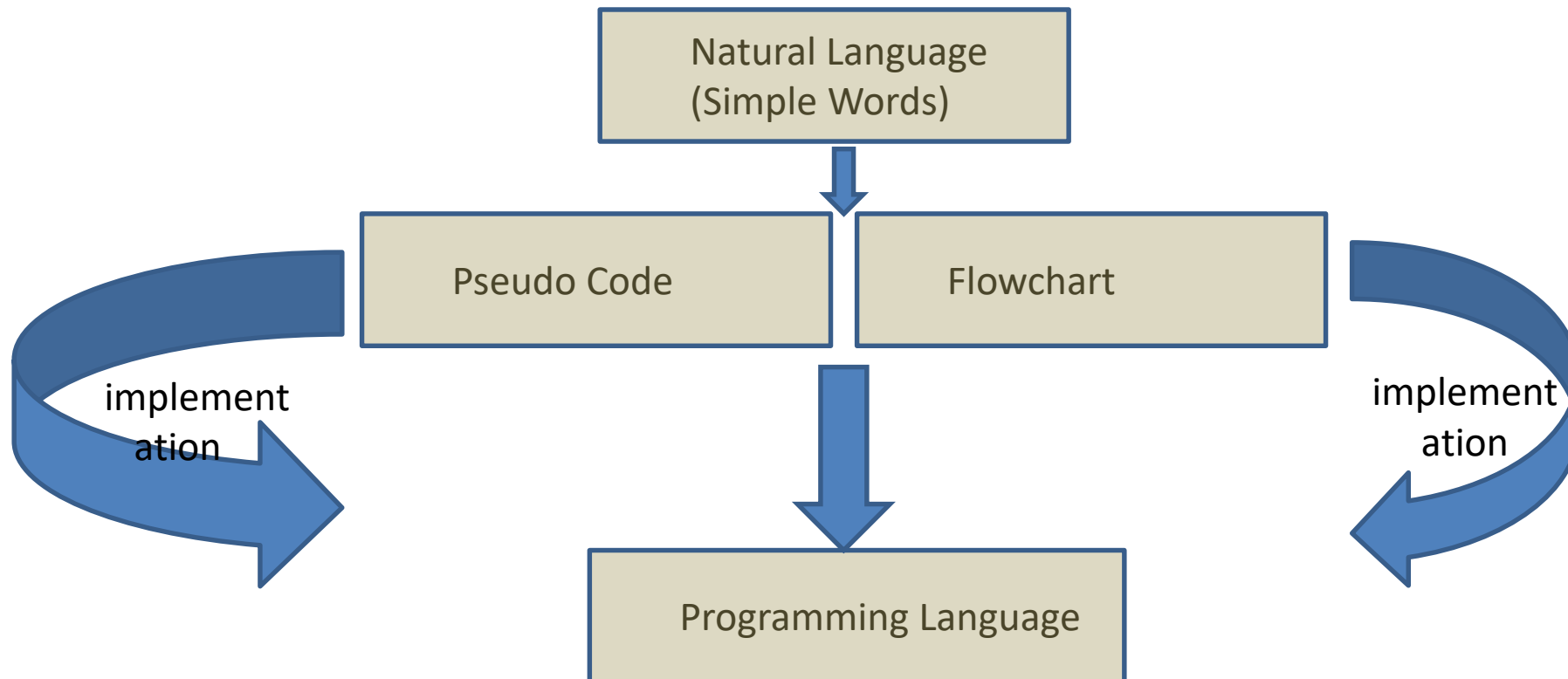
top-down divides a complex problem into multiple smaller parts or modules. Modules are further decomposed until be understood and can not be further decomposed. After achieving a certain level of modularity, the decomposition of modules is ceased.



Top Down Design is also called Modular Programming

Methods of Specifying an Algorithm

Algorithms can be expressed in many kinds of notations, including:



Methods of Specifying an Algorithm

PROBLEM:

Write an algorithm that calculates the sum of two numbers and displays the result.

Natural Language

1. Read A and B
2. Compute the sum, A plus B
3. Print the sum

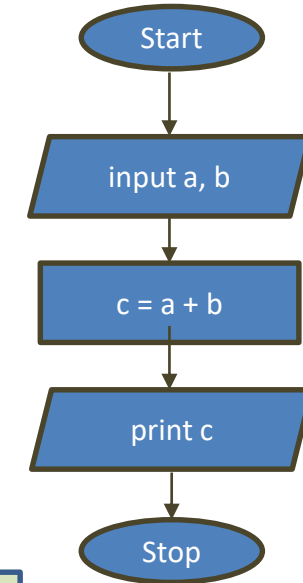
Pseudocode

1. READ a, b
2. COMPUTE $c = a + b$
3. PRINT c

Python 3 Language

1. `a = input("Enter a = ");` # a is a string
2. `b = input("Enter b = ");` # b is a string
3. `c = float(a) + float(b);` # c is float
4. `print("The sum is ", c);`

Flowchart



Pseudocode & python Code

Arithmetic Operators:

+	(sum)
-	(subtraction)
*	(only symbol for multiplication)
/	(division)
=	(assignment)
**	(exponentiation)
()	to associate operations (can't use it for multiplication)
%	(division modulus; remainder of integer division)

Data Types:

- Integer, e.g., 3, 5, -7, 2017
- Real, e.g., 5.3, -7.9, 4.3333, 1e-9
- Char (character), e.g., 'x', 'm', 'u'
- String, e.g., 'John Doe', 'year 2017'
- Boolean, e.g., True or False

Sample of Library Functions:

abs(x), sin(x), arcsin(x), log(x), log10(x), exp(x)

math

pseudocode/code

Example: $x = (y^3 + 2y) \left(\frac{2y^2}{3} \right) \Rightarrow \Rightarrow x = (y^{**3} + 2*y) * (2*y^{**2}/3)$

Problem Solving Methodology:

- At its core, computer programming is

solving problems

Small, medium, large, or very very large
problems; easy, medium, hard problems

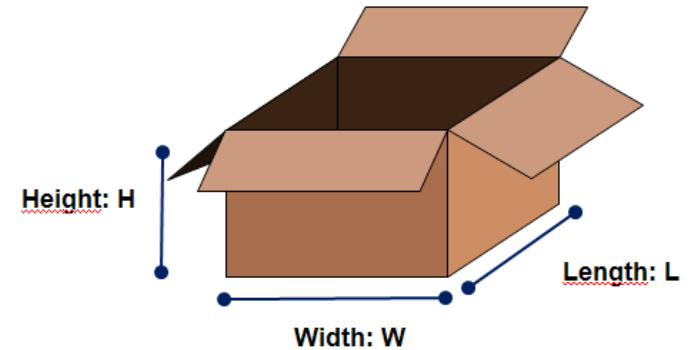
- A **structured methodology** can be use to
construct solutions for a given problem



Some Methodology on Problem Solving:

- We will trace the following sample problem through each of the steps of this problem solving methodology:

Given the 3 dimensions of a box (length, width, and height), calculate the volume.



- The first step to solving any problem is to **decompose** the problem description.
- A good way to do this would be to perform **syntactic analysis** on the description.
- We can do this in **five steps**.



STEPS

Given the 3 dimensions of a box (length, width, and height),
calculate the volume.

Step#1: Identify the **nouns** in the sentence

- Nouns identify information related to data (input) or results (output). Draw a table with 3 columns

Step #2: **Eliminate** redundant or irrelevant information

Step#3: Identify all **verbs** in the sentence

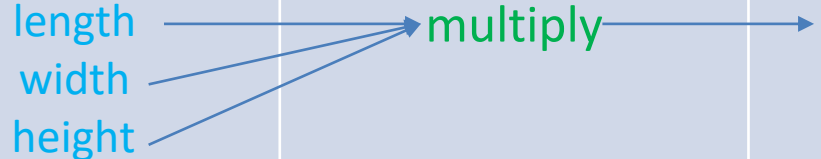
- The verbs identify **actions** your program will need to take.
- Actions, known as **processing** are the steps between your input and your output.

Step#4: Link inputs, processes, and output

Step#5: Use **external knowledge** to complete your solution

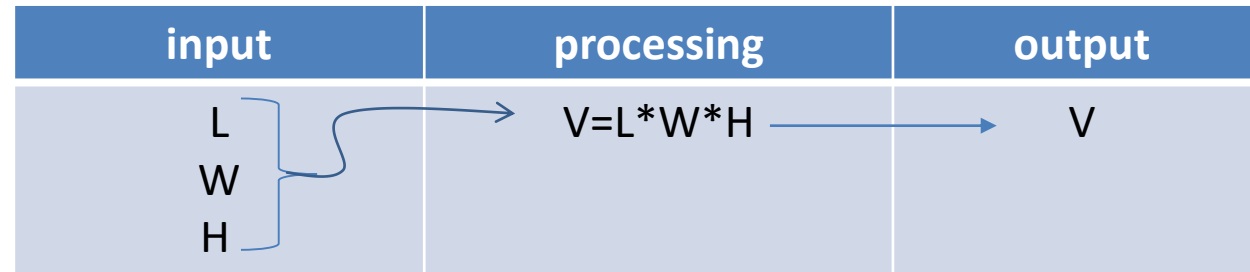
Link

input	processing	output
dimensions length width height	Calculate: multiply	volume



final IPO diagram

Another version of the IPO diagram using variable notation:



L=length
W=width
H=height
V=volume

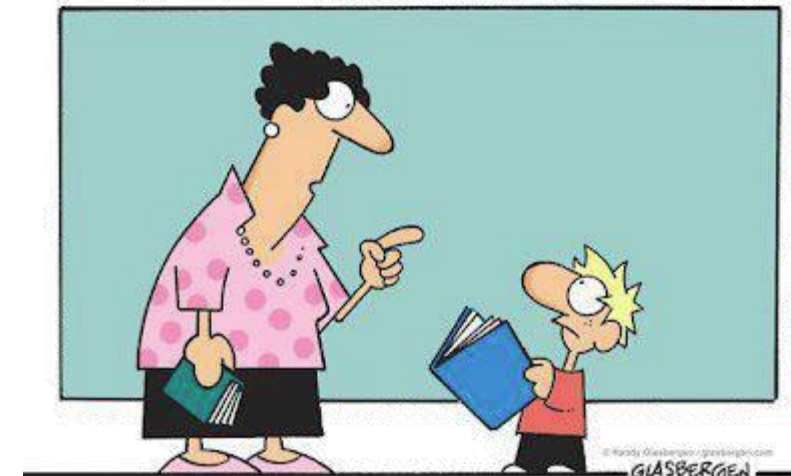
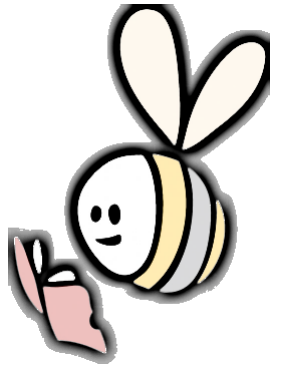
This is more engineering like

variable notation is also known as variable definition

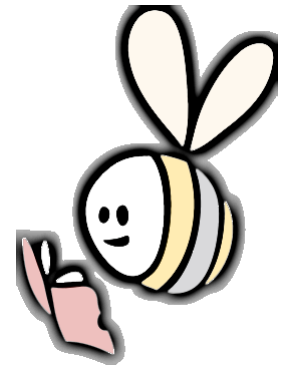
Exercises:

Use the same **DECOMPOSITION** methodology described here to construct IPO diagrams for the following exercises:

- E#1: Converts given Celsius degrees into its equivalent Fahrenheit and Kelvin degrees
- E#2: Compute the final course score with the formula:
 - $\text{score} = \text{cpa}(0.15) + \text{ex1}(0.25) + \text{ex2}(0.25) + \text{final}(0.35)$
 - where cpa is the average of 5 computer projects (cp)
- E#3: Find the product of three numbers. The first number is half the second number. The third number is 3 times the second number.



It's called **reading**.
It's how people install new
software into their brains.

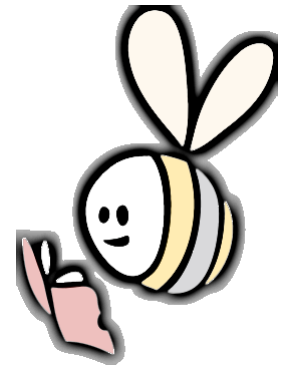


Ex#1: IPO diagram

input	processing	output
Celsius (C)	Compute: $F = (9/5) * C + 32$ $K = C + 273.15$	Fahrenheit (F) Kelvin (K)

E#1: **Converts** given **Celsius degrees** into its equivalent **Fahrenheit and Kelvin degree**

Ex#2: IPO diagram

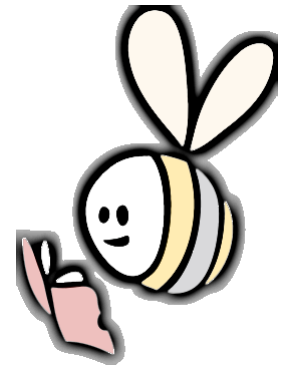


input	processing	output
cp1 cp2 cp3 cp4 cp5	Compute cp Average: $cpa = (cp1 + cp2 + cp3 + cp4 + cp5) / 5$	
ex1 ex2 final	Compute score: $score = cpa(0.15) + ex1(0.25) + ex2(0.25) + final(0.35)$	score

Note how some results from previous calculations are available for new calculations, e.g., cpa

E#2: **Compute** the **final course score** with the **formula**:
 $score = cpa(0.15) + ex1(0.25) + ex2(0.25) + final(0.35)$
where **cpa** is the **average** of **FIVE computer projects**

Ex#3: IPO diagram



input	processing	output
second number, B	Compute 1 st number: $A=B/2$ Compute 3 rd number: $C=3*B$ Compute the product $P=A*B*C$	product of three number, P

E#3: Find the product of three numbers. The first number is half the second number. The third number is 3 times the second number.

- Notice the sequence of operations.
- You need only one number as input for this problem; the other two numbers are actually constructed during processing.
- The results of previous computation, A and C, are data to compute P

Flowcharts

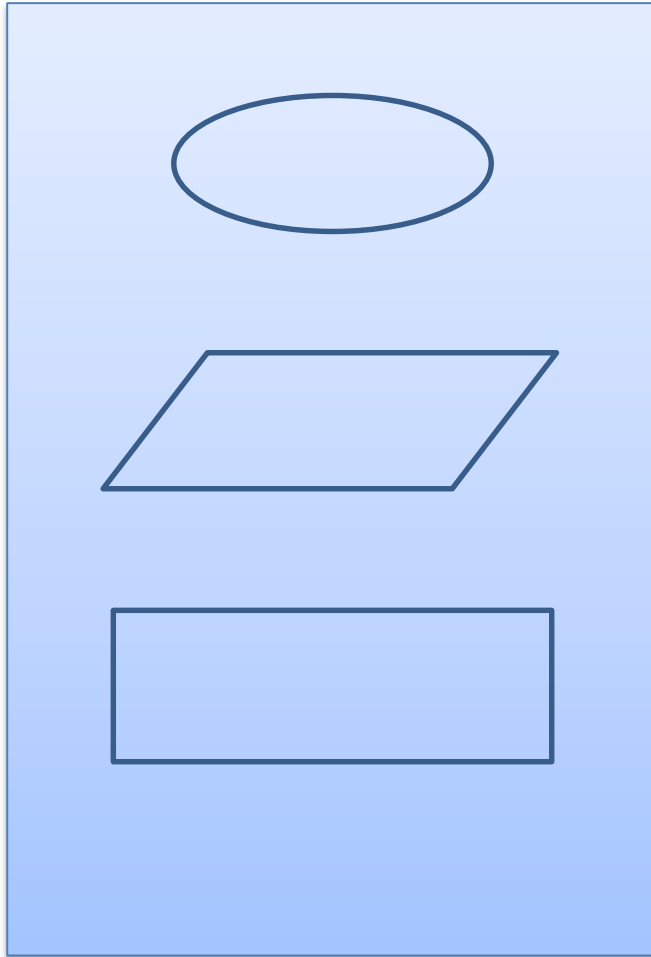
- In the route for developing an algorithm, the second tool for solving our problem involves the use of **flowcharts**.
- A **flowchart** is a **graphical way** of depicting a problem in terms of its inputs, outputs, and processes.
- Each **action** the computer can perform is represented by a different shape. For example, input and output shapes are represented with parallelograms (rhomboids).
- Though the shapes we will use in our flowcharts will be expanded as we cover more topics, the basic elements are next:





Flowcharts are used in many fields to express a process

basic flowchart elements



start/end of a Program (oval)

input/output (rhomboid)

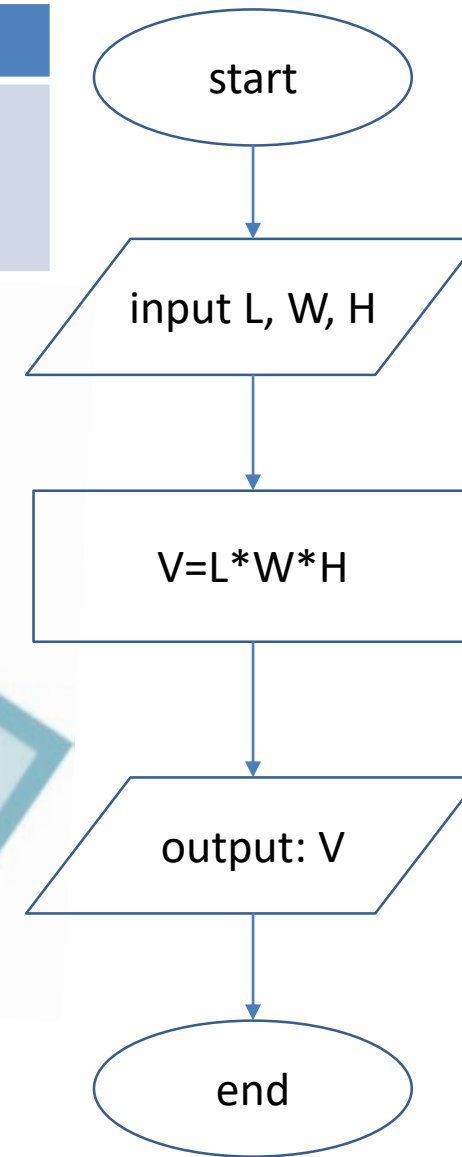
processing (rectangle)

some more elements will be
added later

input	processing	output
L W H	$V = L * W * H$	V

flowchart example

The flowchart may proceed directly from the IPO chart you developed or directly from the problem description



It's a custom to start and end (with symbols) the flowchart of a program

Discuss

- ❖ vertical flowchart
- ❖ one or more actions of the same type in a single shape
- ❖ input/output labels (optional but recommended)
- ❖ arrows to indicate information flow
- ❖ use of math operators vs programming operators

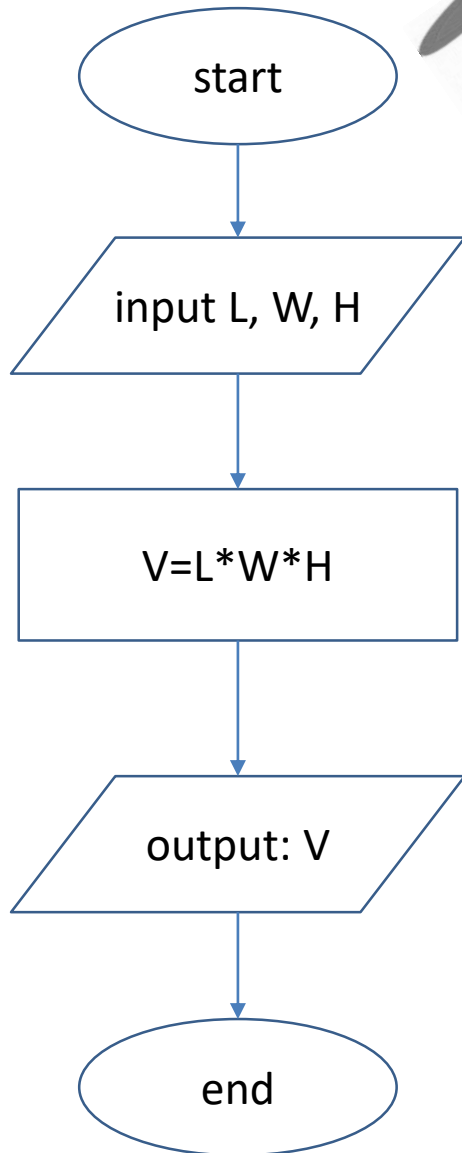
Pseudocode

- The final step in analyzing our problem is to step from our flowchart to construct **pseudocode**.
- Pseudocode involves writing down (with English key words) all of the major steps the program will use to solve a problem.
- Allows you to design a program without having to worry about specific computer language syntax and to concentrate only on the logical steps

Pseudocode

- There are many elements to pseudocode design, only the most rudimentary and essential are described here.
- Keywords that will be used to express program actions:
 - **INPUT**, ASK, READ, GET used to get information from the user (e.g., data)
 - OUTPUT, **PRINT**, WRITE, PUT to report information for the user (e.g., results)
 - **COMPUTE** perform an arithmetic operation
 - **SET** initialize a variable
 - **STORE**, SAVE store a piece of information for later use (e.g., results written in external file)
- All keywords in one row are synonym. Keywords In red are the ones preferred by the instructor
- In this course we use capital letter keywords to differentiate them from high-level language program keywords. Lowercase will be used for matlab codes.

pseudocode example, three versions:



INPUT length, width, height
COMPUTE volume = length * width * height
PRINT volume

INPUT L, W, H
COMPUTE $V = L * W * H$
PRINT V

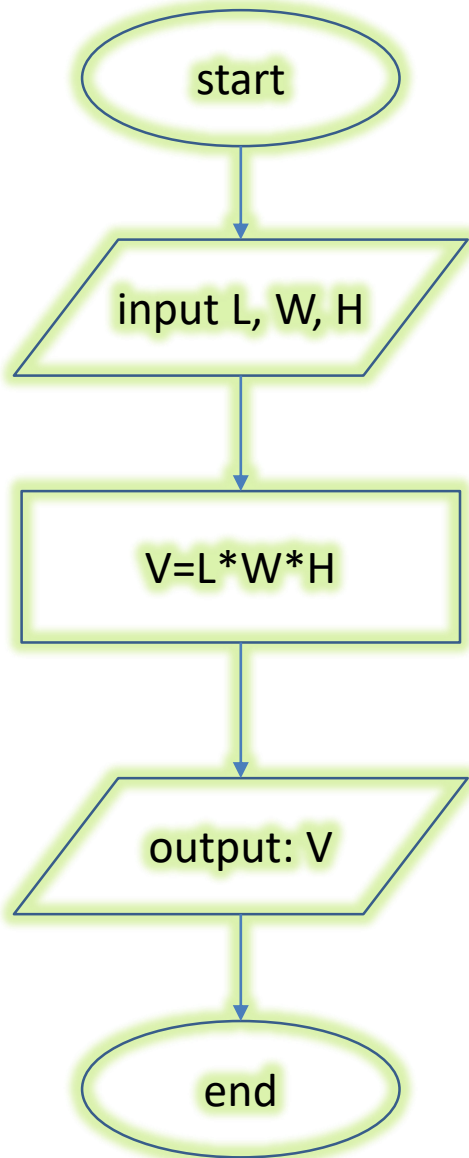
INPUT L, W, H
 $V = L * W * H$
PRINT V

we deliberately eliminated 'calculate' from last pseudocode

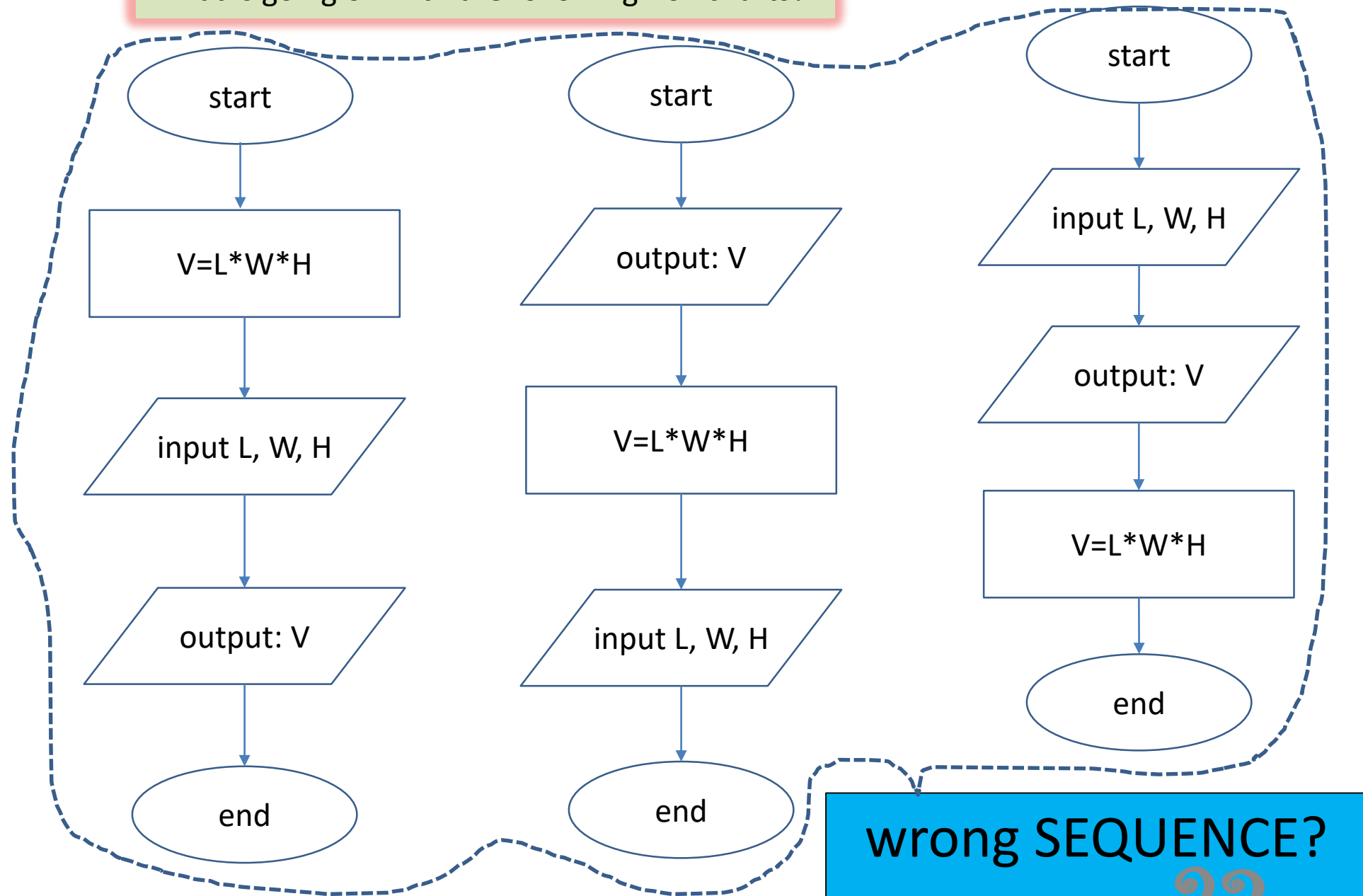
recommended



Correct:



What is going on with the following flowcharts?



A wrong sequence
produces a logic error

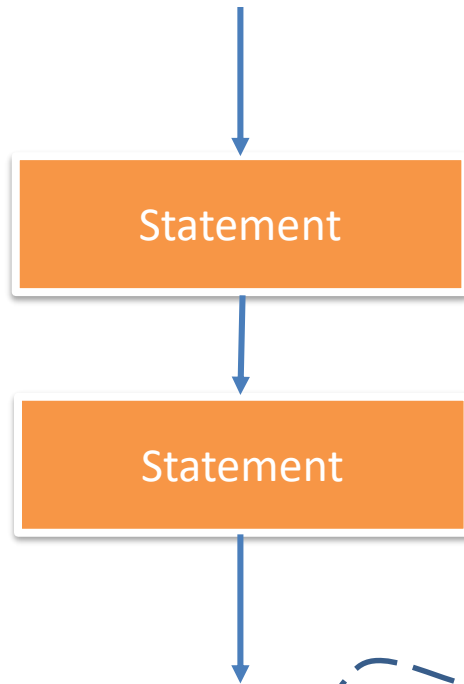


Sequential control is indicated by one action (shape) after another in a flowchart or one statement after another in a pseudocode, each action on a line by itself, and all actions aligned with the same indent (see 2 previous slides and pay attention to the sequence of statements). Then, ask yourself what happens if the sequence was wrong

SEQUENCE (STEPPING)

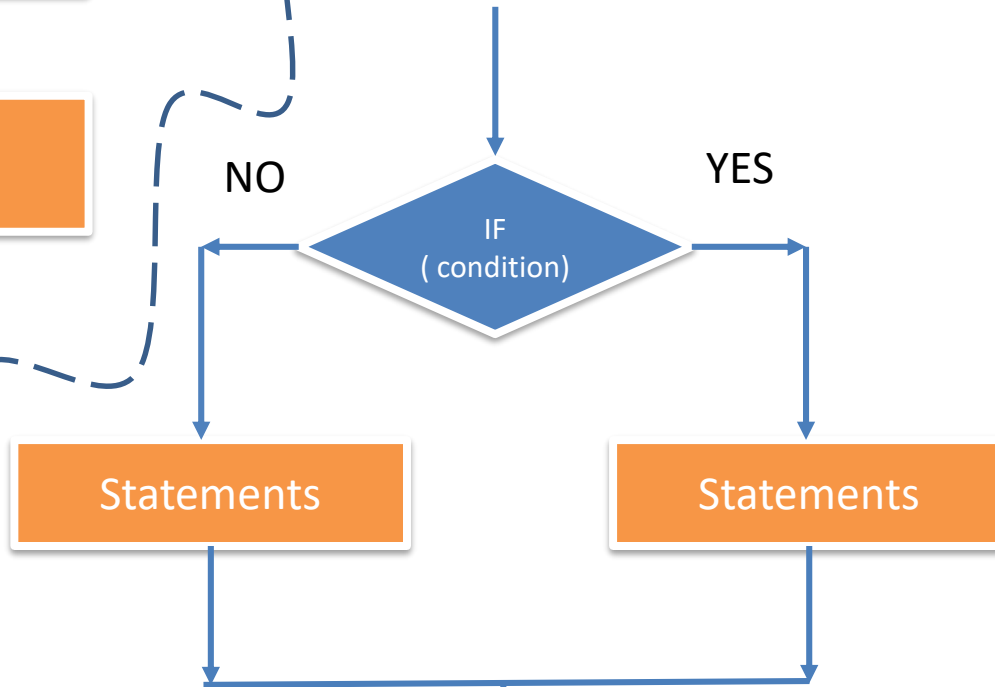
There are other programming structures:

Sequence

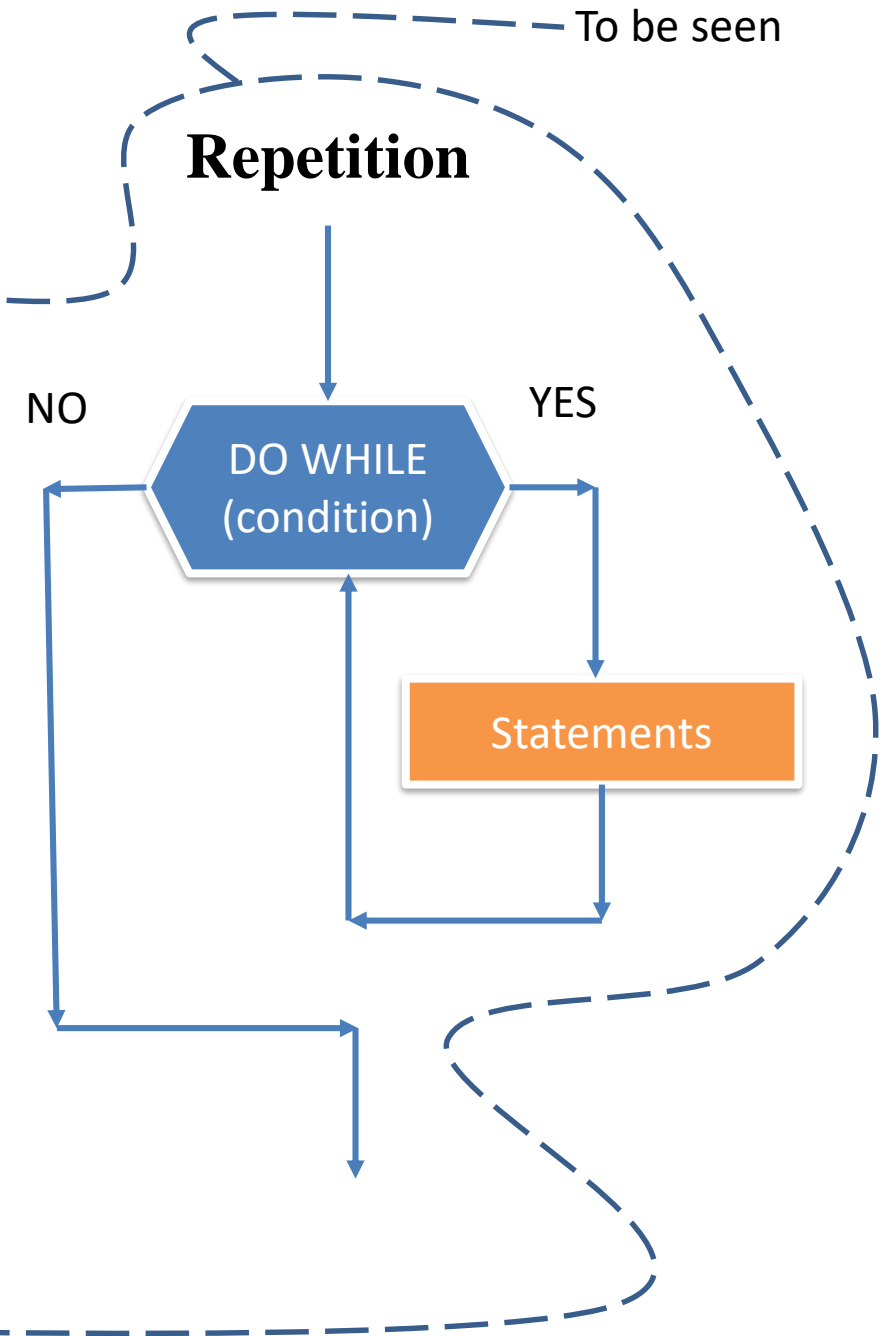


Logic Structures

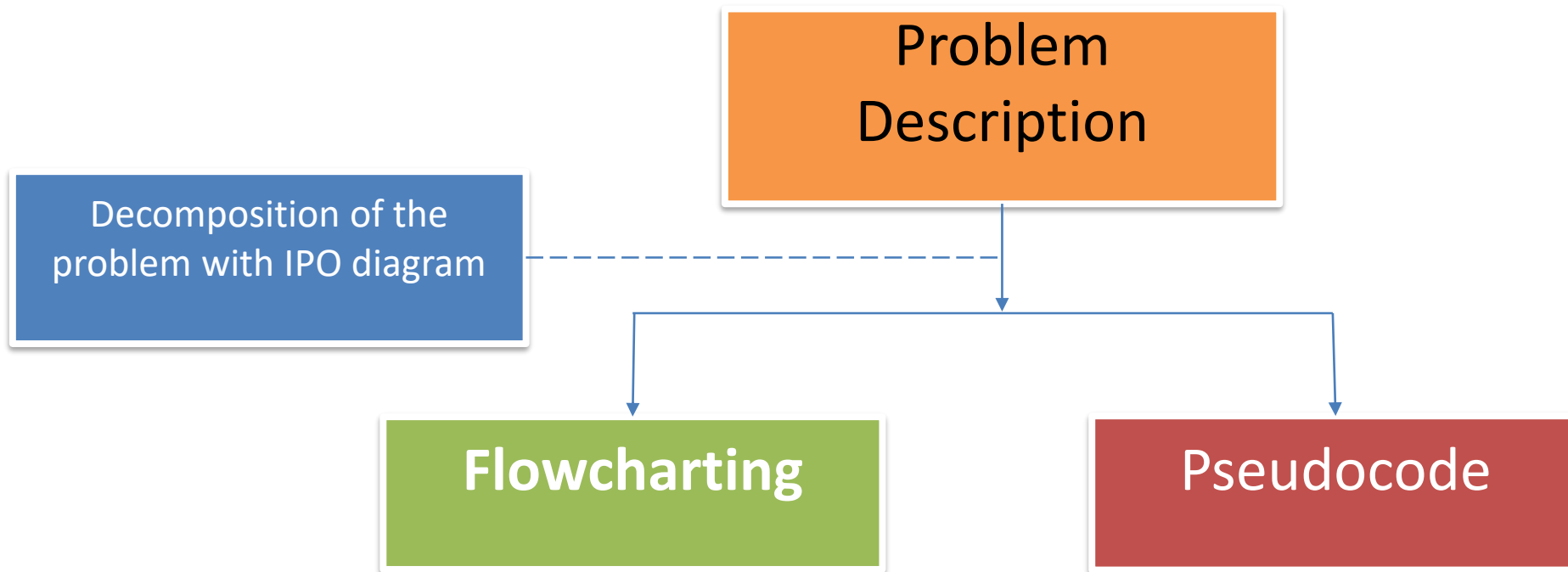
Selection



Repetition



Synthesis of Tools for Program Design & Problem Solving



The Algorithm can be developed starting with the problem description, followed by IPO diagram, followed by flowcharting or pseudocode. Can also be developed directly from the problem description to either a flowchart or pseudocode

Your are ready to solve problems in Sequence Section of “Simple Exercises.pdf”
Look for them in pythongeeks.weebly.com

THE END