Introduction of Computers, Logic and Structure

- Programming Paradigm
- Programming Process: Problem Understanding, Planning, Coding, Translation, and Testing
- Data Hierarchy, Flowchart Symbols, Pseudo Code Statements and Connector
- Variables, Data Types and Evolution of Programming Techniques
- Unstructured Spaghetti Code and Recognizing Structure
- Storage Classes and Basic Structures

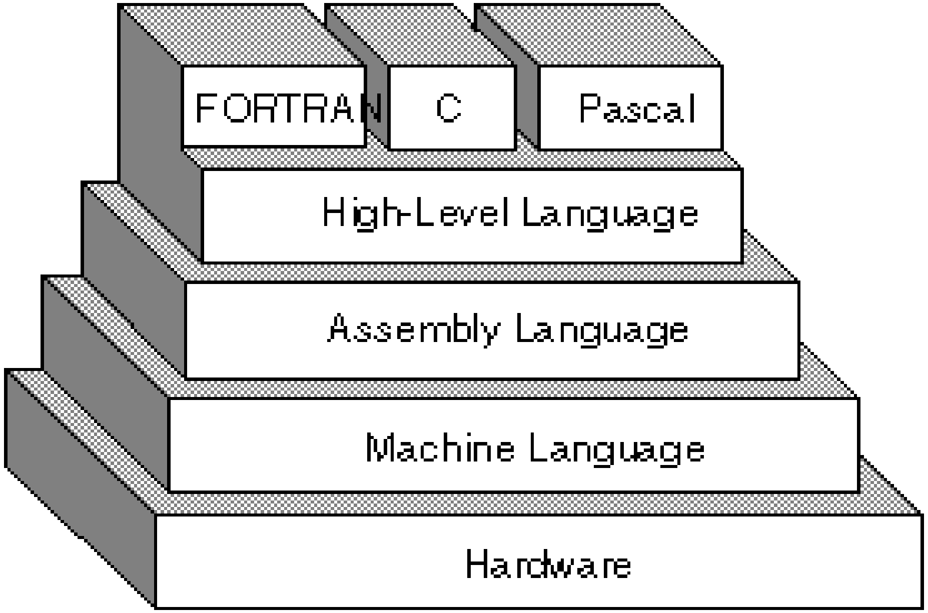
Programming Paradigm

- A programming paradigm is a style or “way” of programming.
- A paradigm is a way of doing something (like programming), and not a real / physical thing (like a language).

Programming Language

- A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.
- The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL,FORTRAN, and Pascal.
- Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

Programming Language (Cont.)



High-level programming languages are simple compared to human languages, are more complex than the languages the computer actually understands, called **machine languages**.
Each different type of CPU has its own unique machine language.

Programming Language (Cont.)

- Lying between machine languages and high-level languages are languages called **assembly languages**.
- Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers.
- Machine languages consist of numbers only.

Programming Language (Cont.)

- Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it. There are two ways to do this:

1) **compile** the program

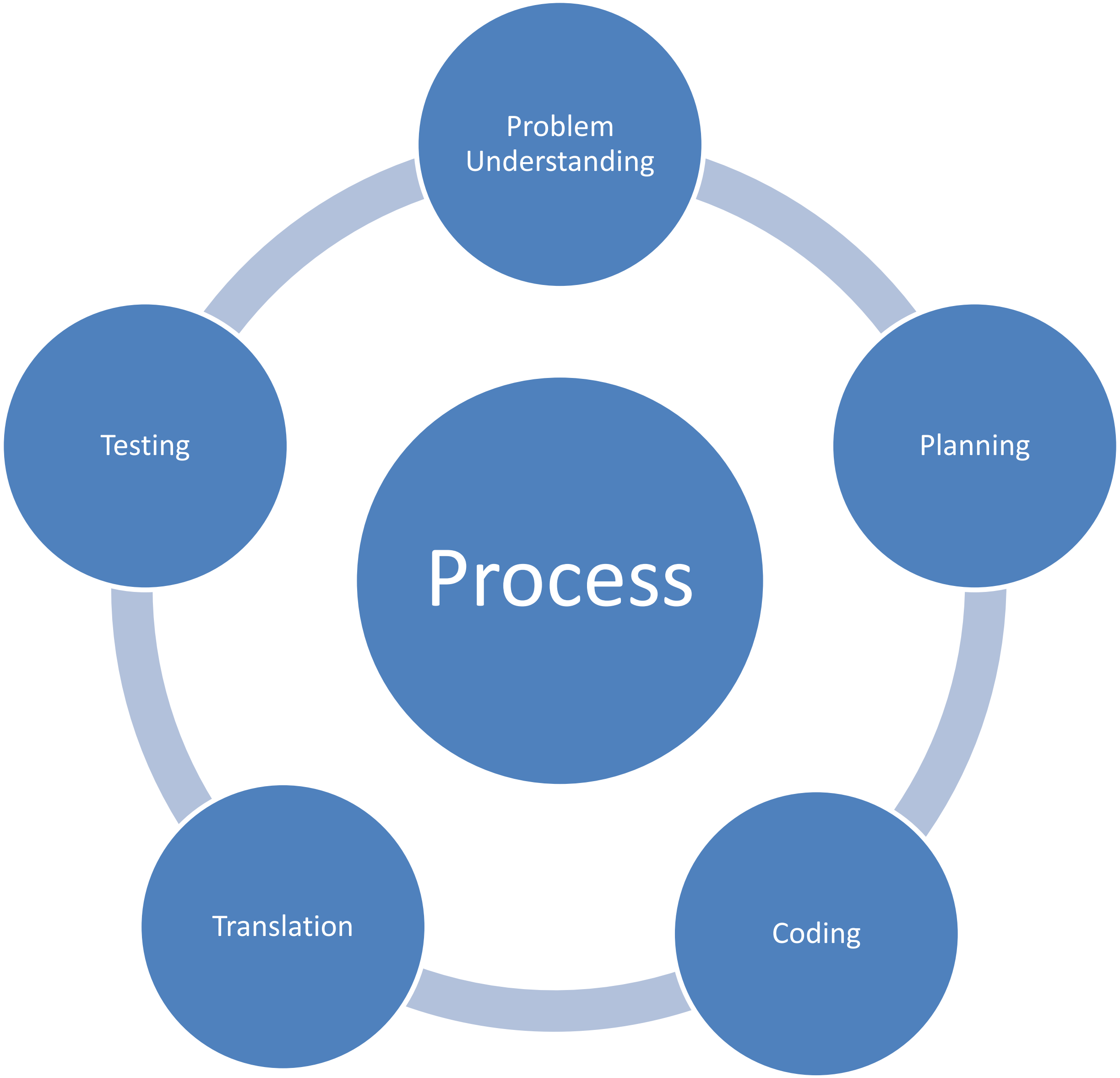
2) **interpret** the program

- **Compiler**
  - The compiler reads a program written in source language(human understandable language) and translates it into machine language – the target language.
  - Machine language is the binary language, the language of 0’s and 1’s which computer can understand.
- **Interpreter**
  - The interpreter reads a program line by line and executes each line to generate output. If any error is found in that line, interpreter will exit and will not read further.

C Programming Language

- In 1972 Dennis Ritchie at Bell Labs writes **C**.
- In 1978 the publication of The **C Programming Language** by Kernighan & Ritchie caused a revolution in the computing world.

Programming Process



Programming Process (Cont.)

- **Understand the problem**
- The programmer should understand the problem thoroughly in terms of requirements i.e. what is the input and output operations to be performed

Programming Process (Cont.)

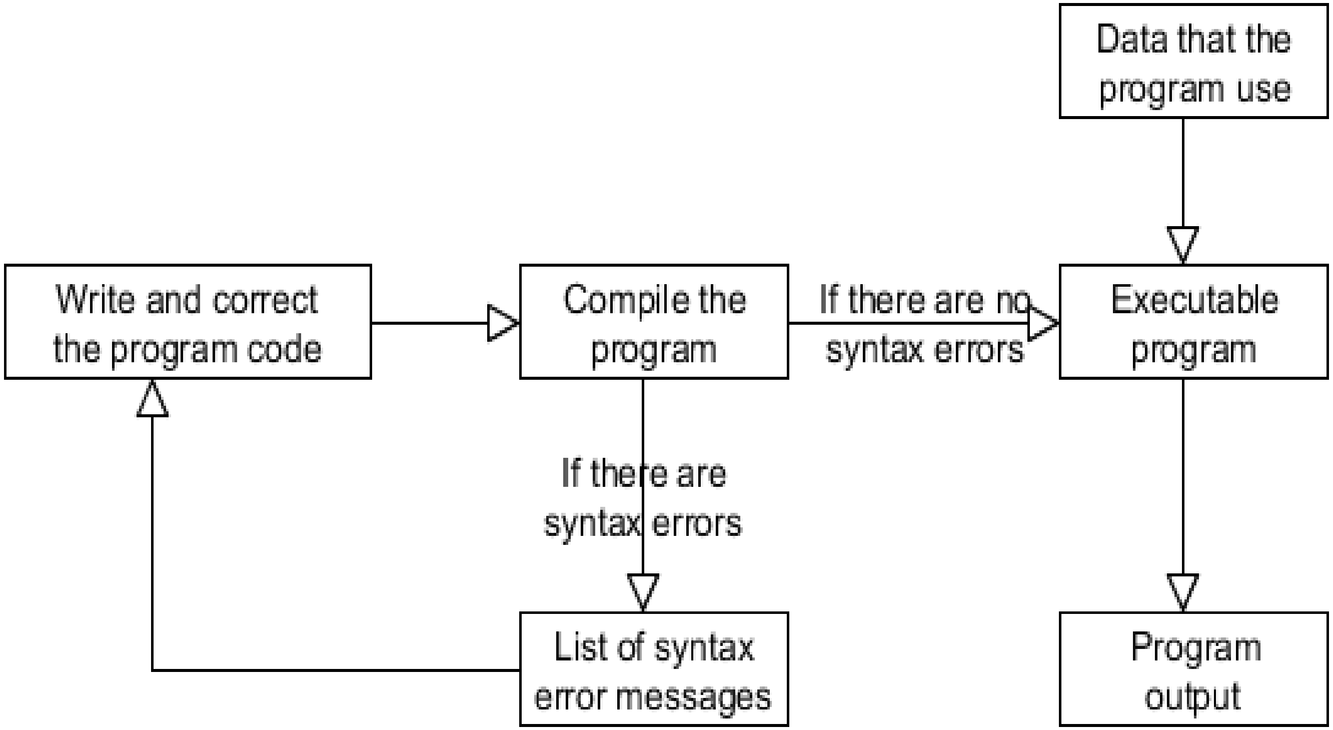
- **Planning the logic**
- Programmer plans the steps of the program, deciding what steps to include and how to order them.
- Thinking carefully about all the possible data values a program might encounter and how you want the program to handle each scenario.
- Algorithm and Flowchart

Programming Process (Cont.)

- **Coding the program**
- Coding is the act of translating the design into an actual program, written in some form of programming language.
- This is the step where you actually have to sit down at the computer and type!
- The logic developed to solve a problem can be executed using any number of languages.

Programming Process (Cont.)

- Using software to **Translate** the program into Machine Language
- Compiler & Interpreter

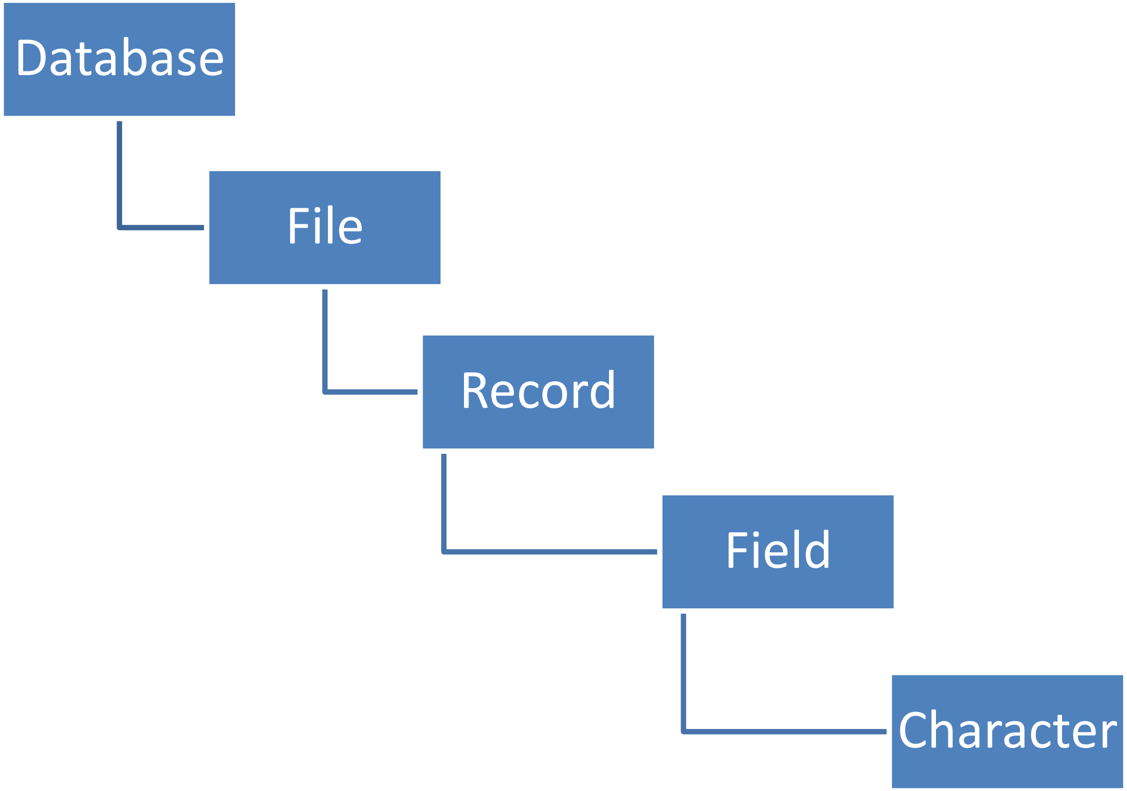

Data that the program use
Write and correct the program code
Compile the program
If there are no syntax errors
Executable program
If there are syntax errors
List of syntax error messages
Program output

Programming Process (Cont.)

- Testing the program
- Program is tested with dummy data.
- Errors in the programming logic are detected during this phase and are removed by making necessary changes in either the logic or the program instructions.

Data Hierarchy

- **Data hierarchy** refers to the systematic organization of **data**.

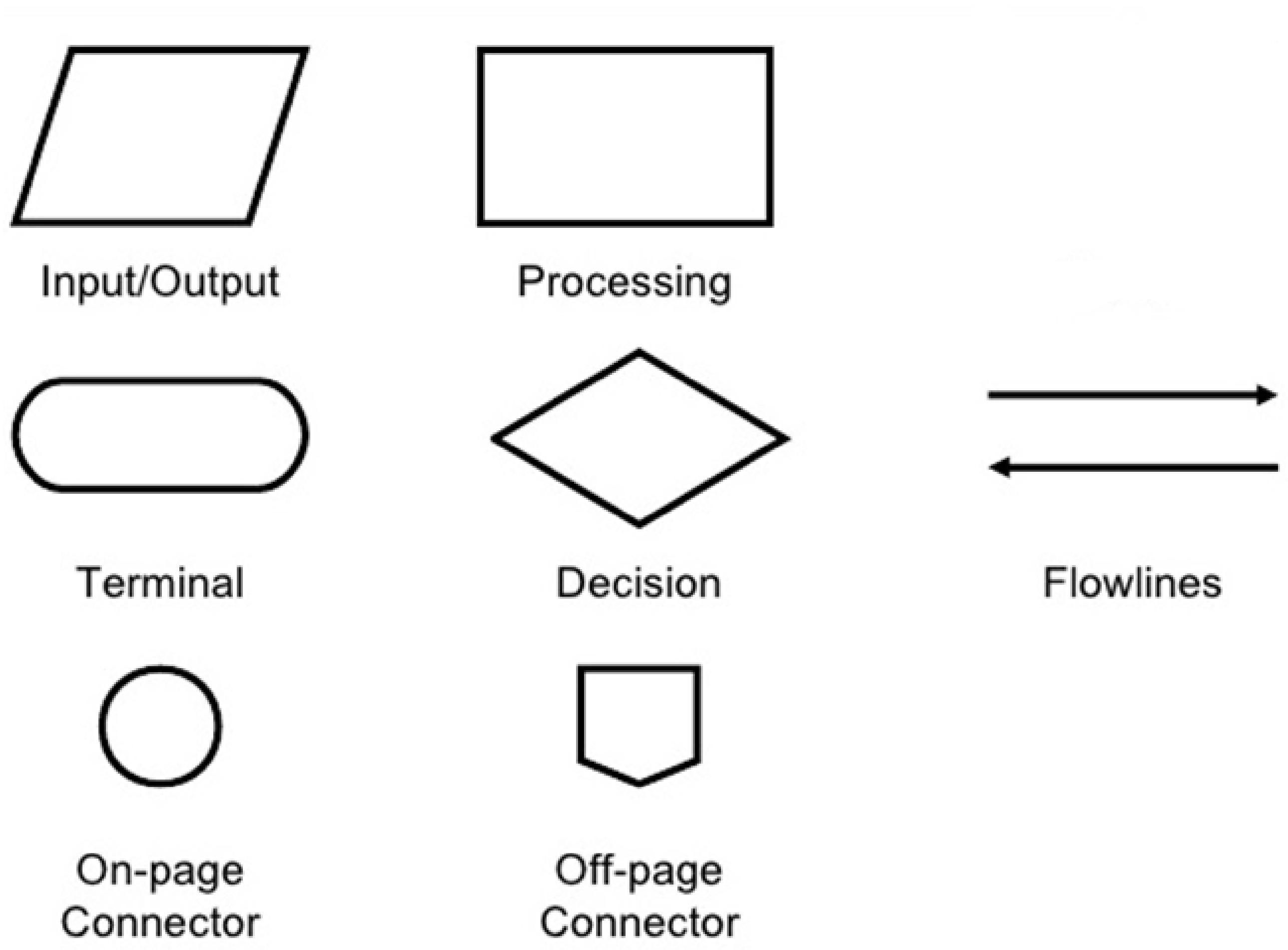


- **Algorithm**
- An algorithm is a step by step instructions require to solve any problem.
- An algorithm is defined as a finite sequence of explicit instructions, which when provided with a set of input values produces an output and then terminates.

- **Flowchart**
- A flowchart is a pictorial representation of the logical steps it takes to solve a problem.
- A flowchart is a pictorial representation of an algorithm in which the steps are drawn in the form of different shapes of boxes and the logical flow is indicated by interconnecting arrows.

- **Pseudo code**
- Pseudo code is a generic way of describing an algorithm without using any specific programming language-related notations.

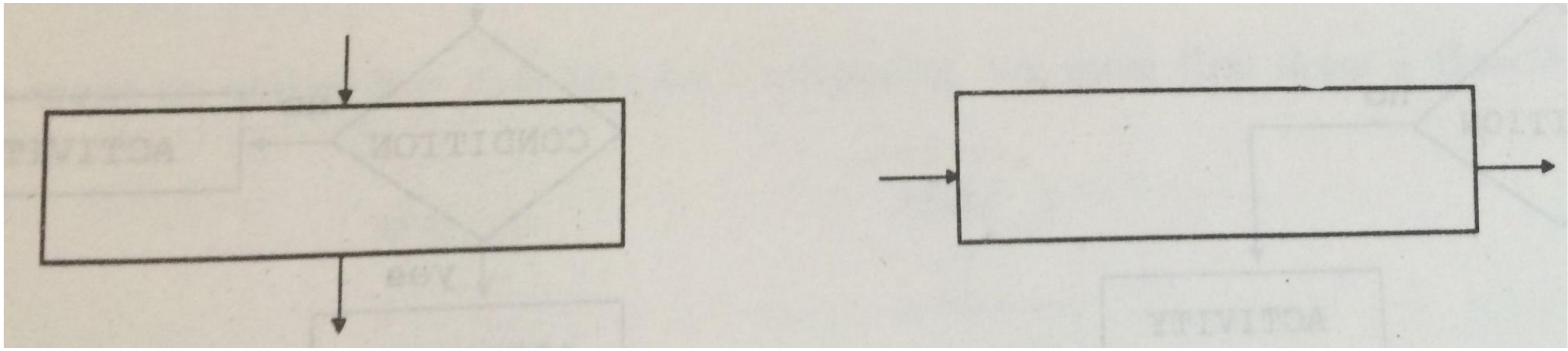
Flowchart Symbols



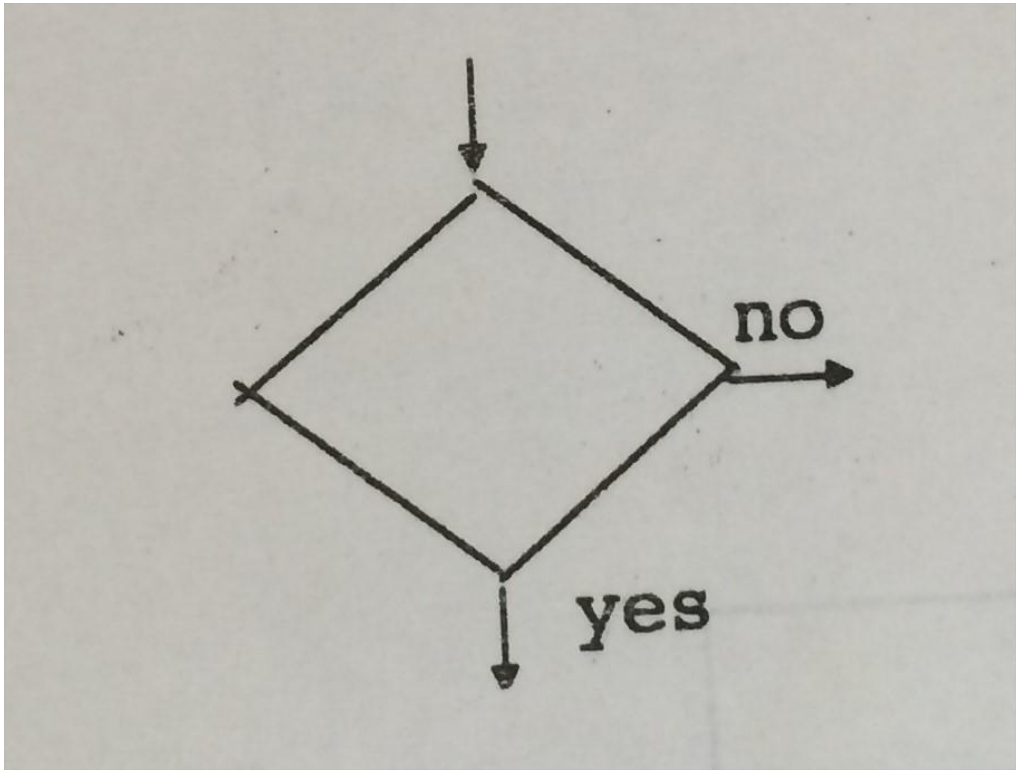
Flowchart Conventions

- The general direction of flow is from top to bottom and from left to right on the page.

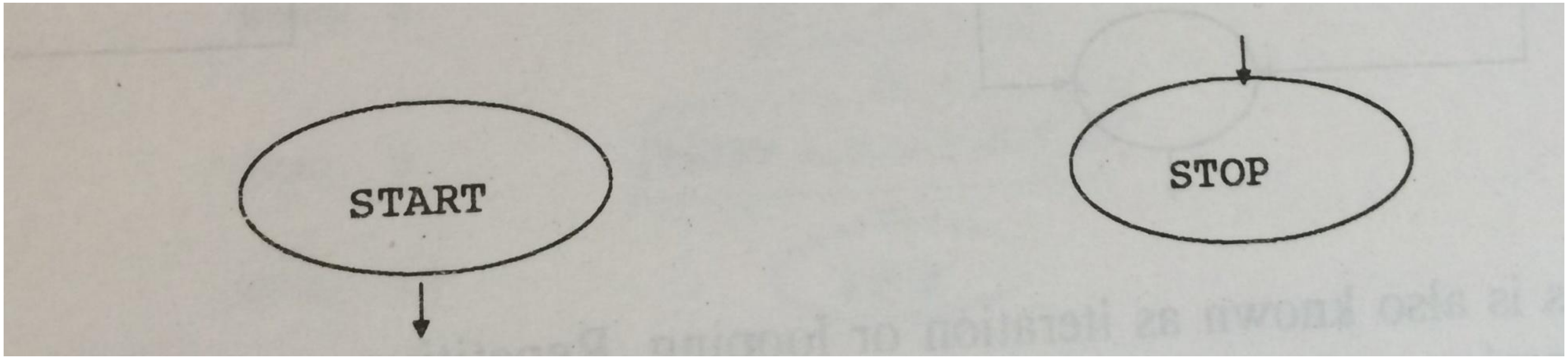
Only one flow line should enter and leave a process symbol.



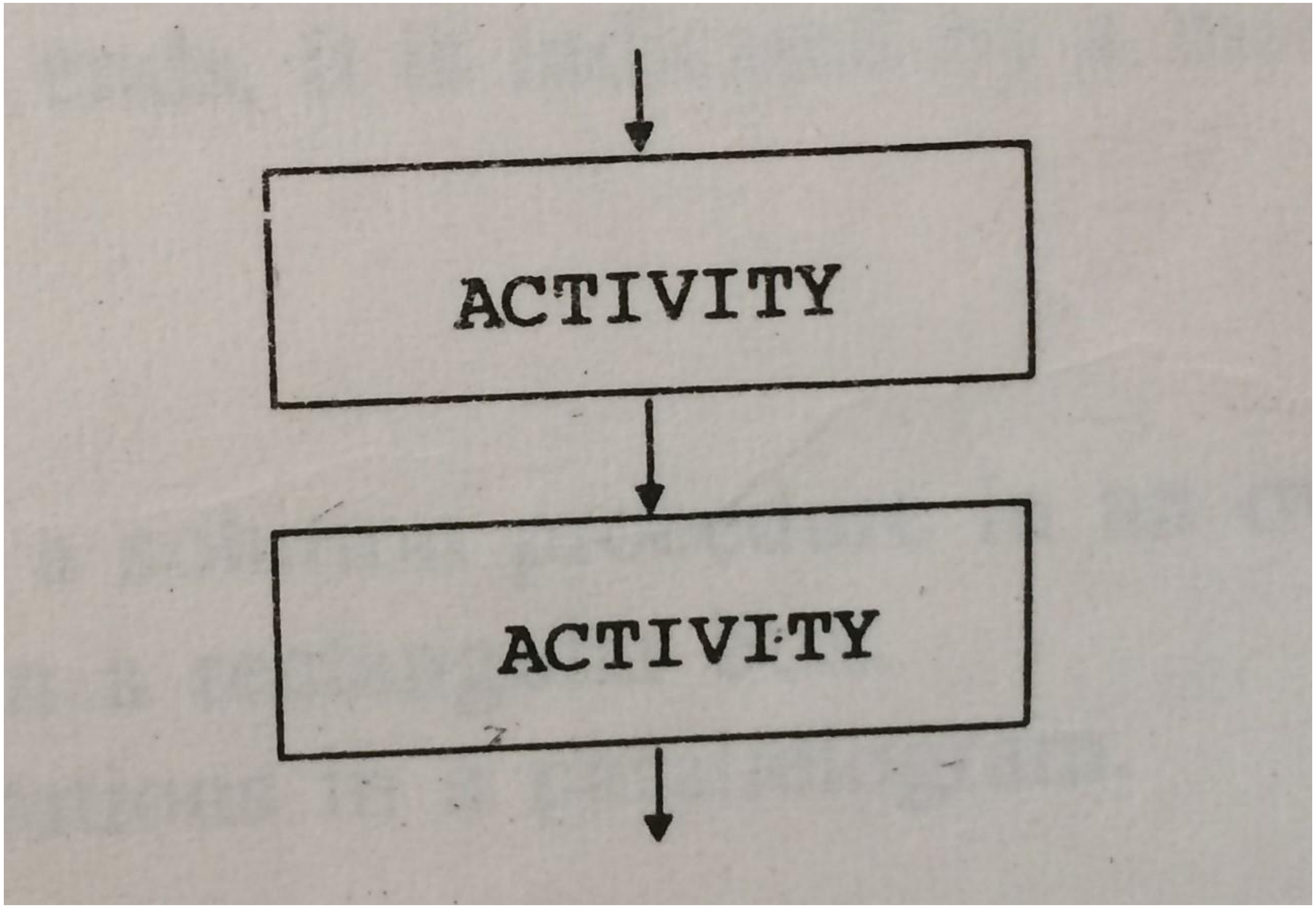
Only one flow line should enter a decision symbol, but two flow lines, one for each possible answer, should leave a decision symbol.



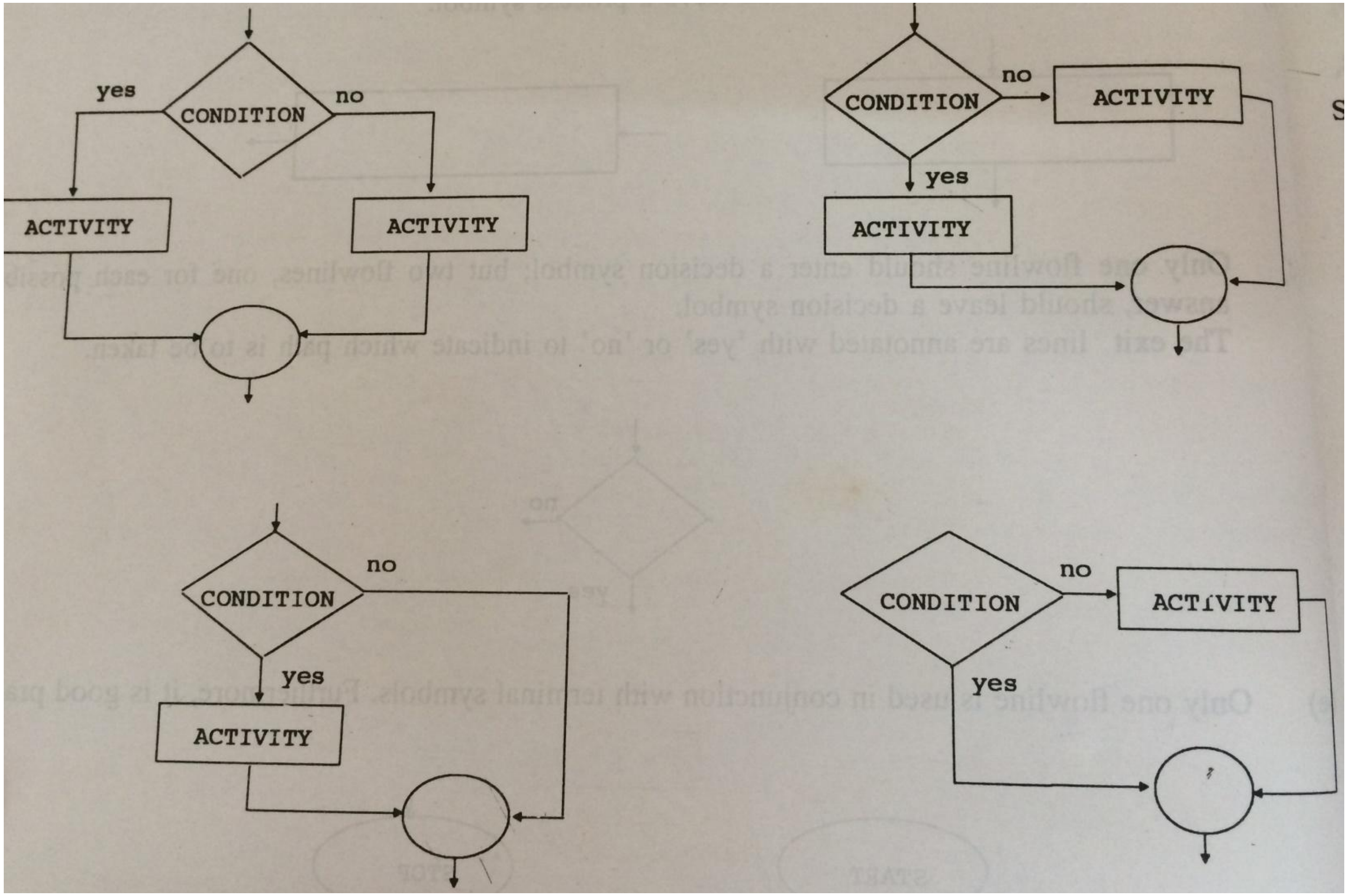
Only one flow line is used in conjunction with terminal symbols.



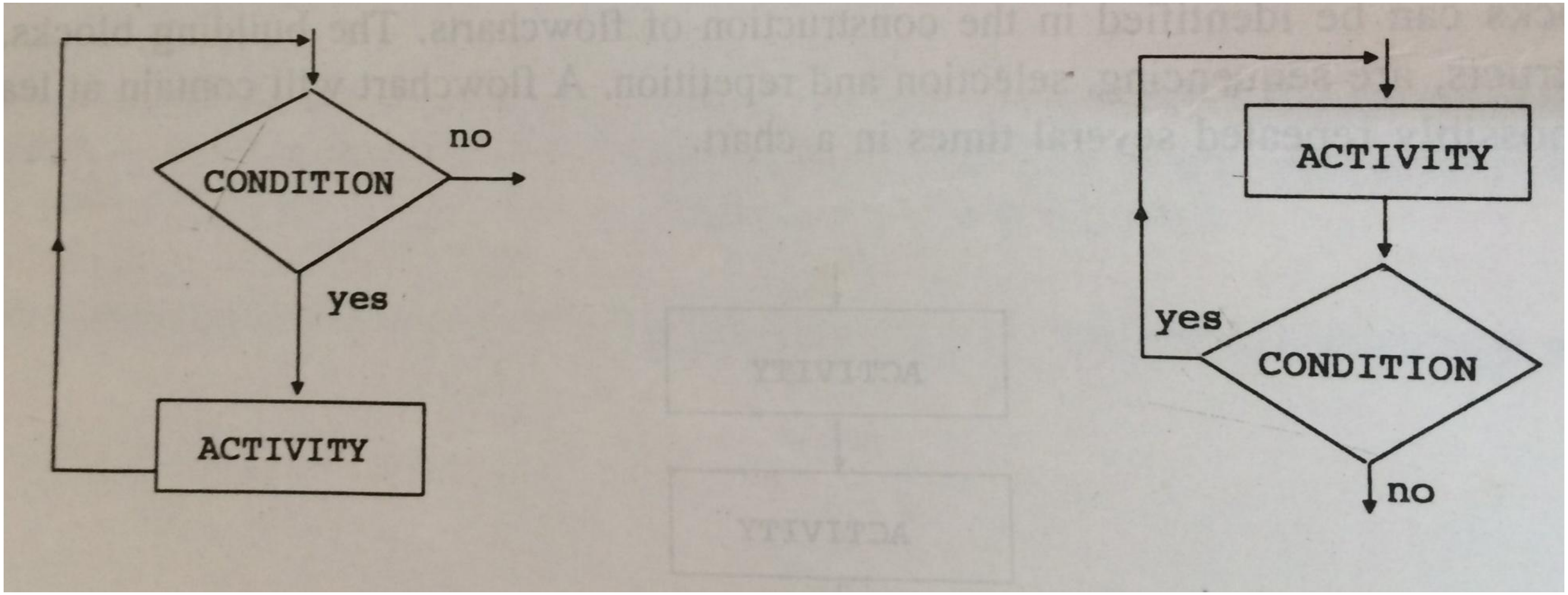
Sequencing



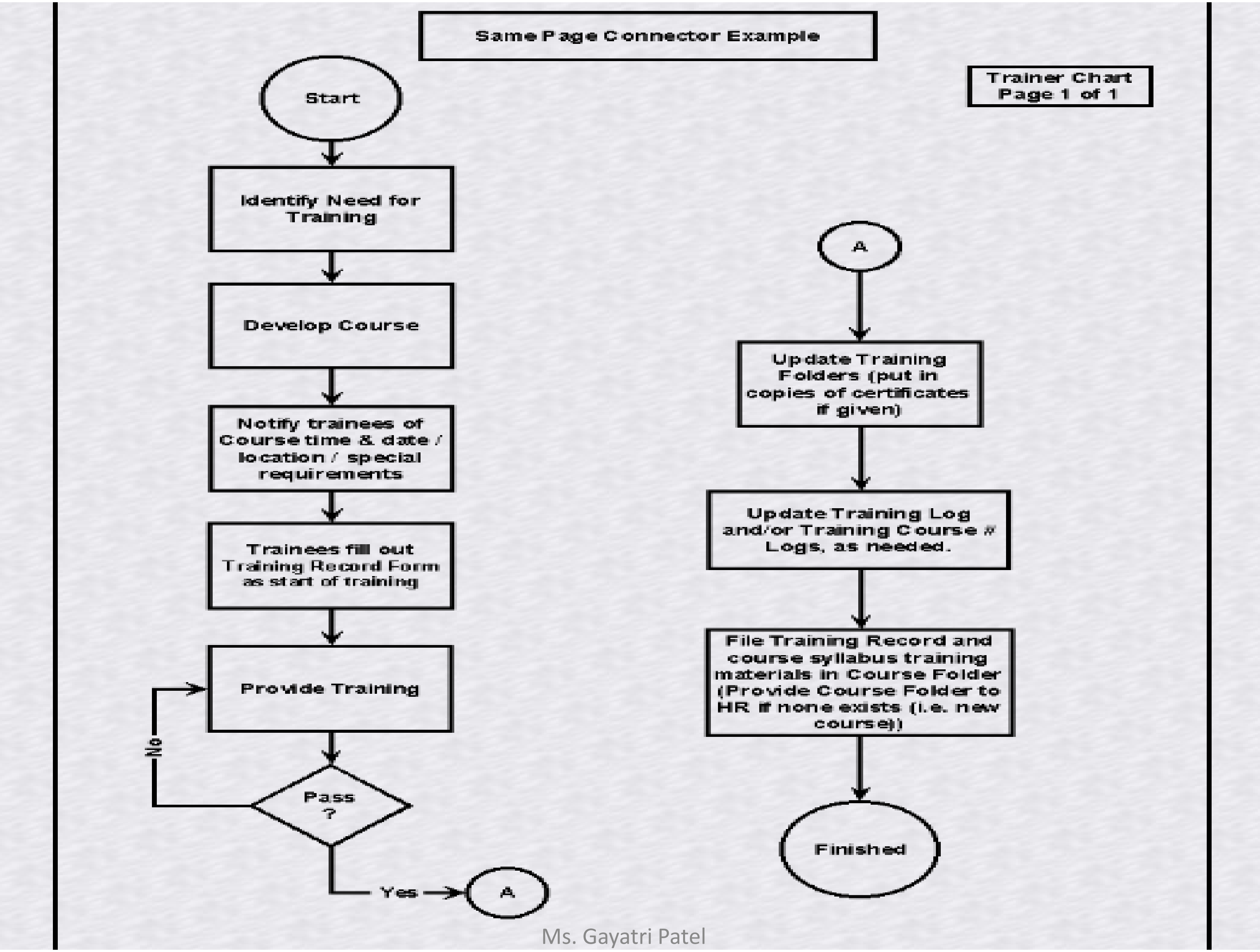
Selection


yes
CONDITION
no
ACTIVITY
ACTIVITY
CONDITION
no
ACTIVITY
yes
ACTIVITY
CONDITION
no
yes
ACTIVITY
CONDITION
no
ACTIVITY
yes

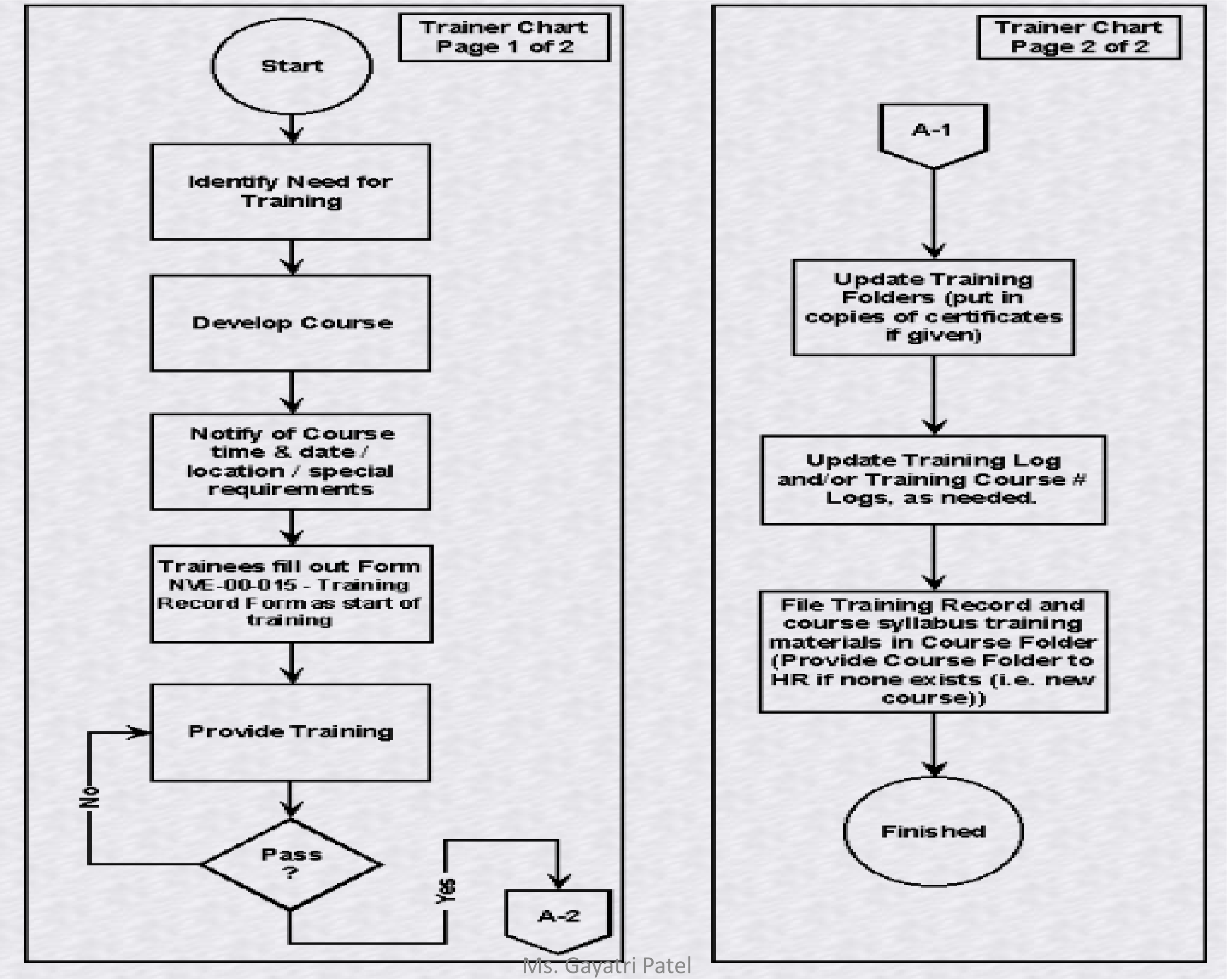
Repetition



On-page Connector


Same Page Connector Example
Trainer Chart
Page 1 of 1
Start
Identify Need for Training
Develop Course
Notify trainees of Course time & date / location / special requirements
Trainees fill out Training Record Form as start of training
Provide Training
No
Pass ?
Yes
A
A
Update Training Folders (put in copies of certificates if given)
Update Training Log and/or Training Course # Logs, as needed.
File Training Record and course syllabus training materials in Course Folder (Provide Course Folder to HR if none exists (i.e. new course))
Finished

Off-page Connector


Trainer Chart
Page 1 of 2
Start
Identify Need for Training
Develop Course
Notify of Course time & date / location / special requirements
Trainees fill out Form NVE-00-015 - Training Record Form as start of training
Provide Training
No
Pass ?
Yes
A-2
Trainer Chart
Page 2 of 2
A-1
Update Training Folders (put in copies of certificates if given)
Update Training Log and/or Training Course # Logs, as needed.
File Training Record and course syllabus training materials in Course Folder (Provide Course Folder to HR if none exists (i.e. new course))
Finished

Data Types

- A type defines a set of values and a set of operations that can be applied on those values.
- C Language has defined a set of types that can be divided into 4 general categories:
- Void
- Integral
- Floating-point
- Derived

Data Type (Cont.)

Category	Type	Purpose	Size (Bytes)	Range of values
Void	void	Has no values and no operations.		
Integral	boolean	Can represent the Boolean values: true or false. nonzero number – true and zero number – false.		
	char	For storing characters and strings.	1 Byte	-128 to +127
	int	For storing integers.	2 Byte	-32768 to +32767
Floating-point	float	For storing real numbers.	4 Byte	3.4e-38 to 3.4e+38
	double	For storing double precision floating numbers.	8 Byte	1.7e-308 to 1.7e+308

Variable

- A memory storage object whose value can be changed during the execution of a program.
- A variable is a data name that may be used to store a data value.

- Variable declaration
- int i;
- Variable initialization
- i = 5;

Examples of Variable

- char code = ‘b’;
- int myint = 14;
- float f = 14.3;
- double pi = 3.1415926536;
- int i, sum = 30,min,max;
- i=5;

Variable conventions

- A name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program.
- Variable names may consist of letters, digits, and underscore (_).
- They must begin with a letter.
- Some systems permit underscore as the first character.
- Length 30 characters.
- Uppercase and lowercase are significant. That is, the variable **Total** is not the same as **total** or **TOTAL**.
- Space is not allowed.

Example

- John
- Delhi
- mark
- Value
- x1
- T_raise
- Ph_value
- distance

- 123
- (area)
- %per
- 25th
- Price$
- Group one
- Average_number

Unstructured Spaghetti Code

- **Spaghetti code** is a phrase for source code that has a complex and tangled control structure, especially one using many GOTO statements, exceptions, threads, or other "unstructured" branching constructs.
- It is named such because program flow is conceptually like a bowl of spaghetti, i.e. twisted and tangled.
- Spaghetti code can be caused by several factors, such as continuous modifications by several people over a long life cycle.

Storage Classes

- Storage classes specify the scope of objects.
- Type qualifiers specify processing limitations on objects.
- Storage class specifier :
  - auto
  - register
  - static
  - extern

Object storage attributes

- Scope
  - Block
  - File
- Extent
  - Automatic
  - Static
  - Dynamic
- Linkage
  - Internal
  - External

Scope

- **Scope** defines the visibility of an object, it defines where an object can be referenced.
- Block
  - The boy of a function is a block and compound statement in a loop is a nested block within its function block.
- File
  - File scope includes the entire source file for a program, including any files included in it.

Extent

- The **extent** of an object defines the duration for which the computer allocates memory for it.
- Automatic
  - An object with an automatic extent is created each time its declaration is encountered and is destroyed each time its block is exited.
- Static
  - A variable with a static extent is created when the program is loaded for execution and is destroyed when the execution stops.
- Dynamic
  - Dynamic extent is created by the program through the *malloc* and its related library functions.

Linkage

- A large application program may be broken into several modules, with each module potentially written by a different programmer.
- Each module is a separate source file with its own objects.
- Different modules may be related when the program is link edited.
- Internal
  - An object with an internal linkage is declared and visible only in one module. Other modules cannot refer to this object.
- External
  - An object with an external linkage is declared in one module but is visible in all other modules that declare it with a special keyword, extern.

Auto variables

- A variable with an auto specification has the following storage characteristic:
- Scope : block
  - The variable must be declared in a block.
- Extent : automatic
  - Each time the declaration is encountered, it is re-created.
- Linkage : internal
  - It is visible only in the source file in which it is declared.

Register variables

- A register storage class is the same as the auto with only one difference.
- The declaration includes a recommendation to the compiler to use a central processing unit (CPU) register for the variable instead of a memory location.
- This is done for efficiency.
- The time required to access CPU register is significantly less than the time required to access a memory location.
- There is one restriction on the use of a register variable, a register variable address is not available to the user.
- This means that we can’t use the address operator and the indirection operator with a register.

Static variables with Block scope

- When it is used with a variable that is declared in a block, static defines the extent of the variable.
- Scope: block
- A static variable in this context can be referred to only in the block it is defined.
- Extent: static
- The computer allocates storage for this variable only once.
- Linkage: internal
- It is not visible in other modules.
- Example
  - static int x = 1;

Static variables with File scope

- When the static specifier is used with a variable that has file scope and we want to keep its linkage internal, it is defined with the specifier static.
- Scope : file
- Extent : static
- Linkage : internal

Extern

- A variable declared with a storage class of extern has a file scope, the extent is static, but the linkage is external.
- Scope: file
- Extent : static
- Linkage : external

- It is common, on large projects, to decompose the project into many source files. The decomposed source files are compiled separately and linked together to form one unit.

Exercise

- The distance between two cities (in KM) is input through the keyboard. Write an algorithm & draw a flowchart to convert & print this distance in meters, feet, inches and centimeters.
- If the marks obtained by a student in five subjects are through the key board, Write an algorithm and draw a flowchart to find out the aggregate marks and percentage marks obtained by the student. Assume the maximum marks that can be obtained by the student in any subject is 100.

- The temperature of the city is input through the keyboard, write an algorithm and draw a flowchart to convert this temperature into centigrade degrees.
- The length & breath of a rectangle and radius of a circle are input through the keyboard. Write an algorithm and draw a flowchart to calculate the area & perimeter of a rectangle, and area & circumference of the circle.

- The basic salary of an employee is input through the keyboard. If the dearness allowance is 40% of basic salary and house rent allowance is 20% of basic. Write an algorithm and draw a flowchart to calculate his gross salary.
- Two numbers, A & B, are input through the keyboard. Write an algorithm and draw a flowchart to evaluate A+B, A-B, A×B, A/B, A^B.

- If principal, number of years, & rate of interest is input through the keyboard, Write an algorithm and draw a flowchart for calculating the simple interest.
- If three numbers are input through the keyboard, write an algorithm and draw a flowchart to pick up the largest of the three numbers.

- The year of joining of an employee and the current year are input through the keyboard. If the years of service are greater than 3 then the employee is given a bonus of Rs. 3000. Write an algorithm and draw a flowchart to print on the screen the bonus only if the years of service are greater than 3.

- If five numbers are input through the keyboard, write an algorithm and draw a flowchart to find out the smallest of the five numbers.

- If temperature of city is input through the keyboard in Fahrenheit degrees, determine how the atmosphere of the city is according to following rules:

If temperature is above 40 degrees centigrade	Very hot
If temperature is between 37 & 40 degrees centigrade	Hot
If temperature is between 34 & 36.9 degrees centigrade	Warm
If temperature is less than 34 degrees centigrade	Cool

- $C = (F - 32) 5/ 9$

- $sum = s1 + s2 + s3 + s4 + s5$
- $per = (sum * 100) / 500$

- Rectangle area = $l * b$
- Rectangle perimeter = $2 (l + b)$
- Circle area = πr^2
- Circle Circumference = $2 \pi r$

- 1 km = 1000 m
- 1 km = 3280.84 feet
- 1 km = 100000 centimeter
- 1 km = 39370.1 inch

- Factorial of 5
- 5! = 5 × 4 × 3 × 2 × 1

  = 120

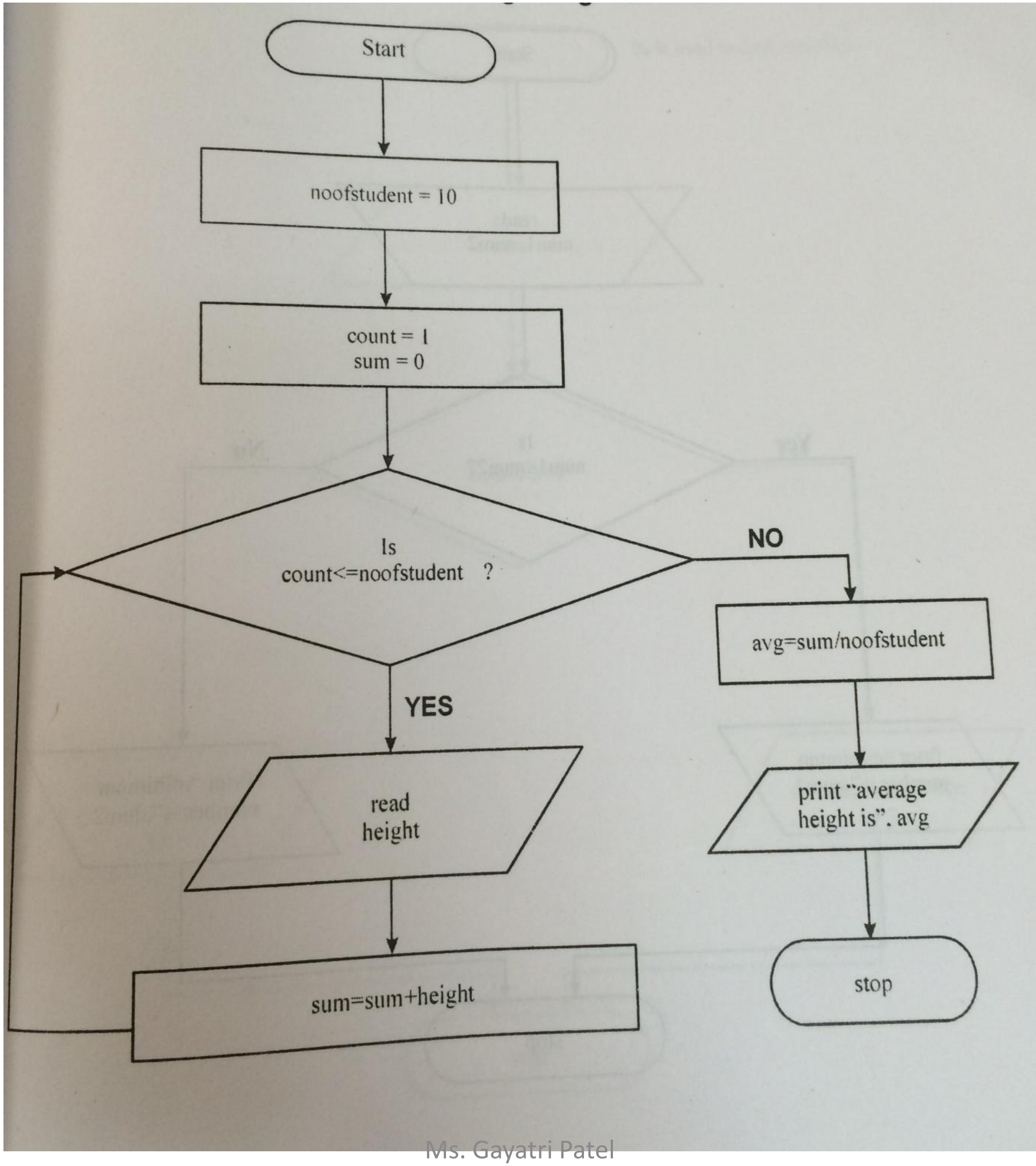
- Area of triangle = ½ bh
- Perimeter of triangle = side1 × side2 × side3

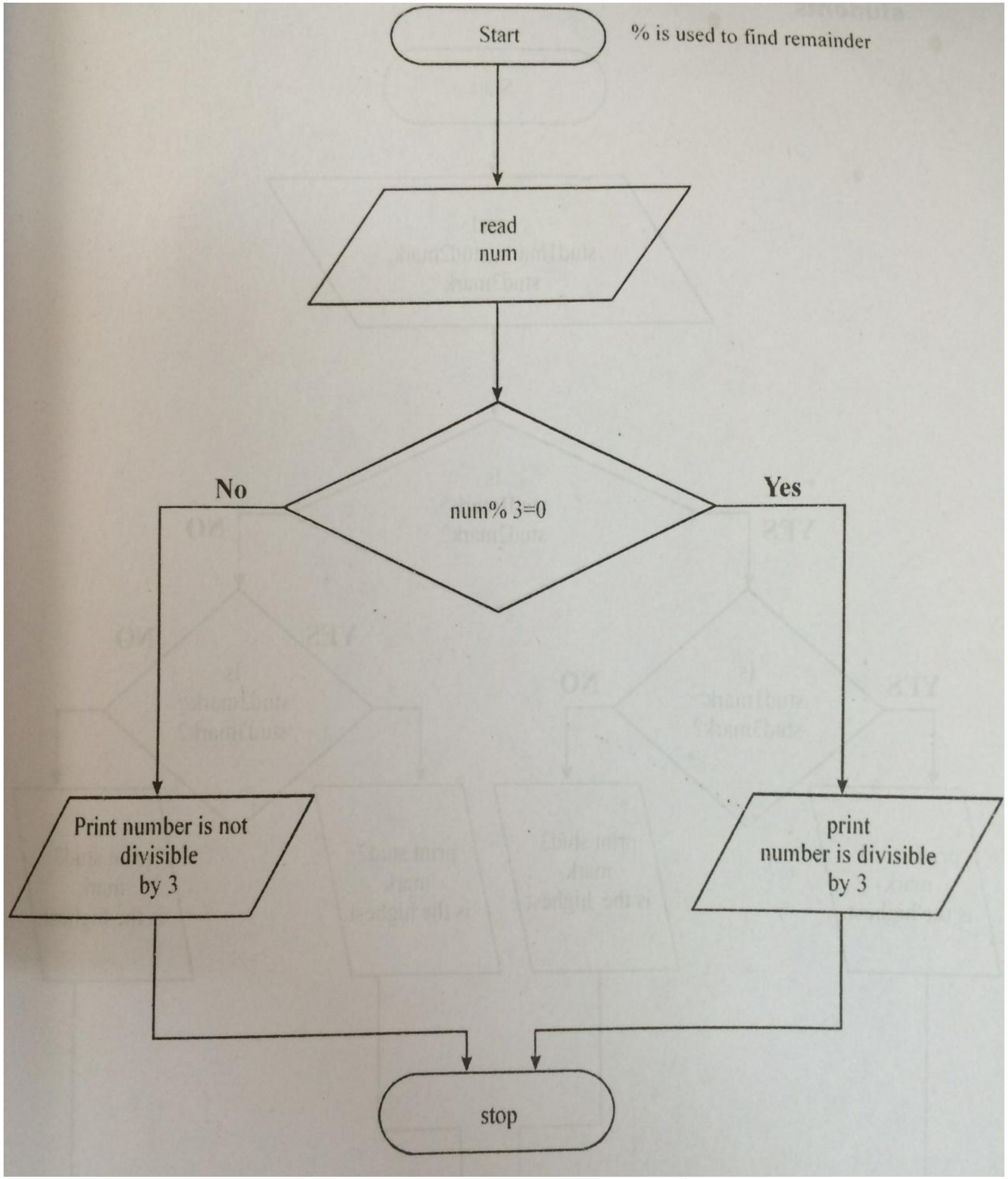
Write an Algorithm and Draw a Flowchart

- To show multiplication of two numbers.
- To find area and perimeter of the triangle.
- To find average height of 10 students.
- To find whether the given number is divisible by 3.
- To find out which has got highest marks out of 3 students.
- To reverse given number.
- To show addition of all the digits of a given number. 543 : 5 + 4 + 3 = 12
- To find factorial of a given number.

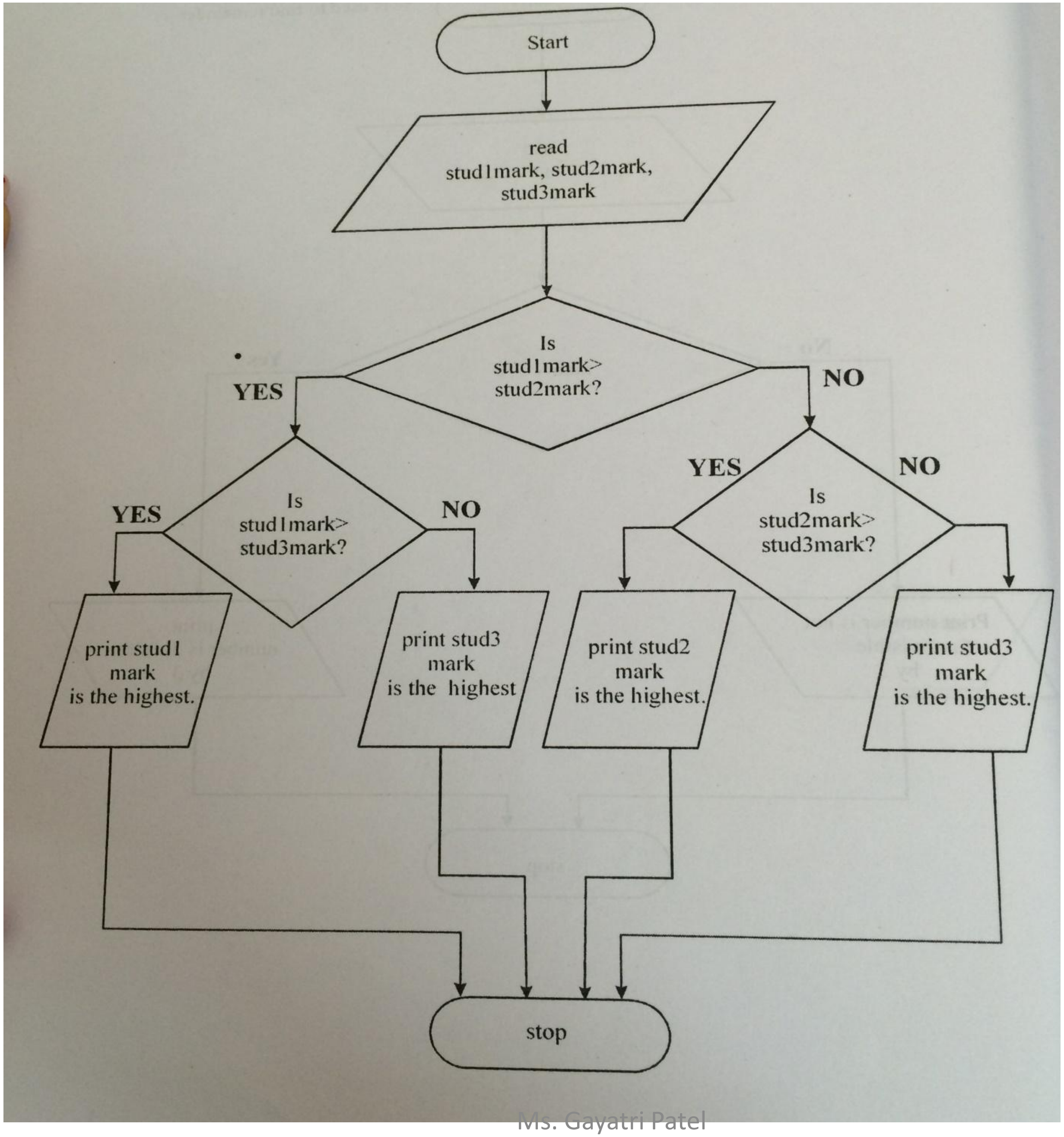
Algorithm for reverse number

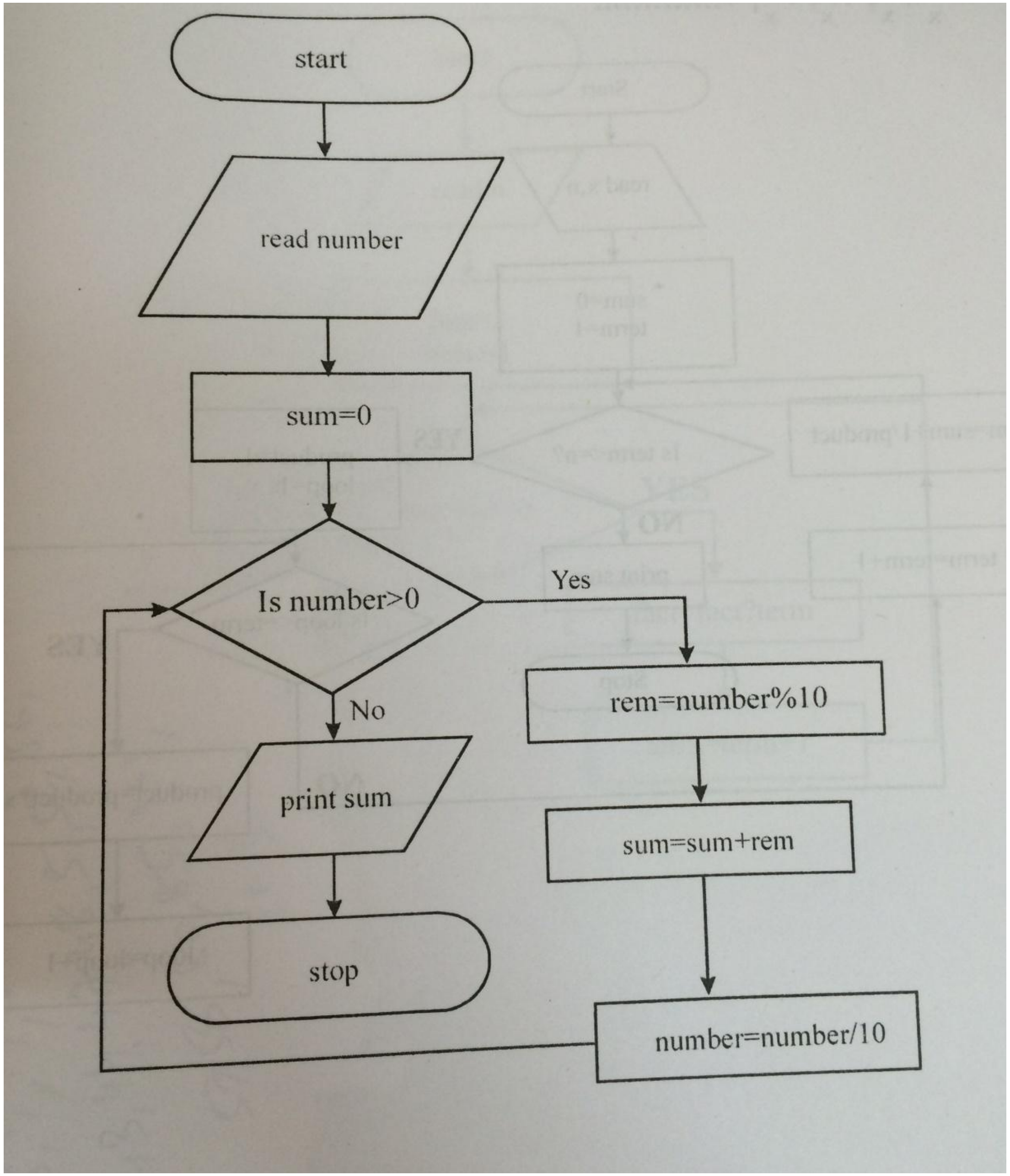
```
Step 1- start
Step 2- input number in num
Step 3- take sum=0
Step 4- repeat from step 4 to step 7 until num>0
Step 5- calculate r=num%10
Step 6- calculate sum=sum*10+r
Step 7-calculate num=num/10
Step 8- print “reverse number=“, sum
Step 9- stop
```

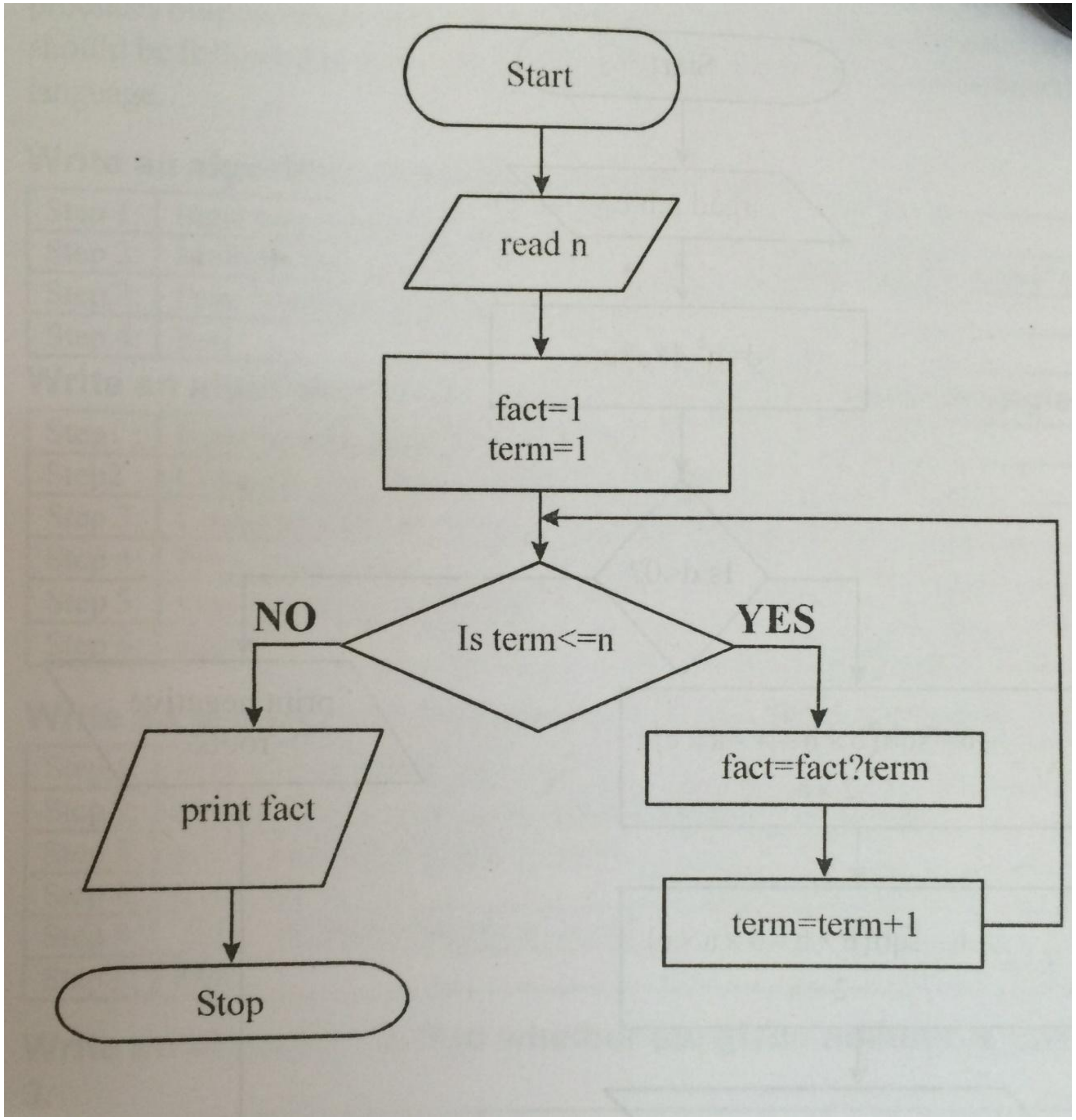

Start
noofstudent = 10
count = 1
sum = 0
Is
count<=noofstudent ?
NO
YES
avg=sum/noofstudent
read
height
print "average
height is". avg
sum=sum+height
stop


Start
read num
num% 3=0
No
Yes
Print number is not divisible by 3
print number is divisible by 3
stop

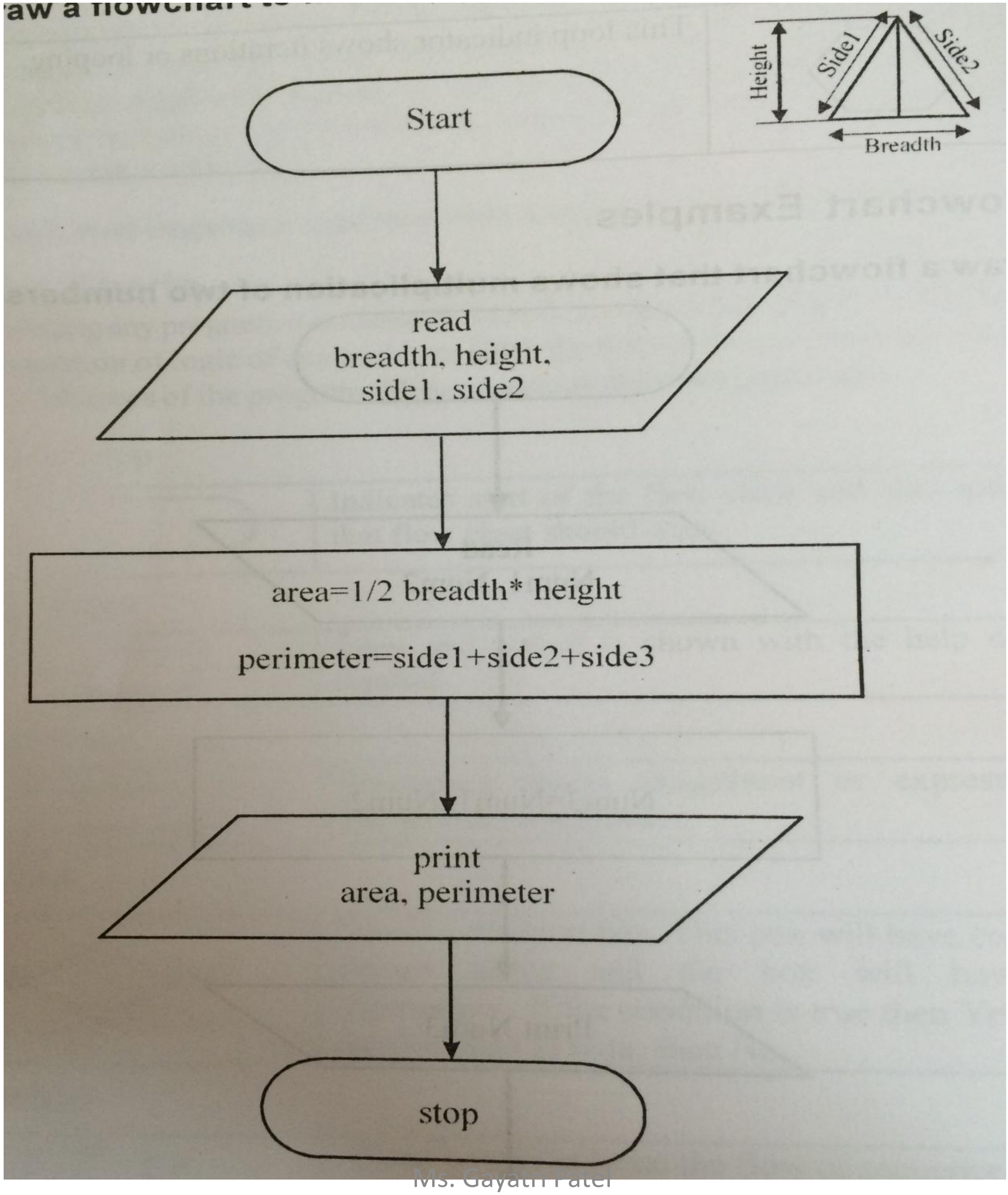
% is used to find remainder

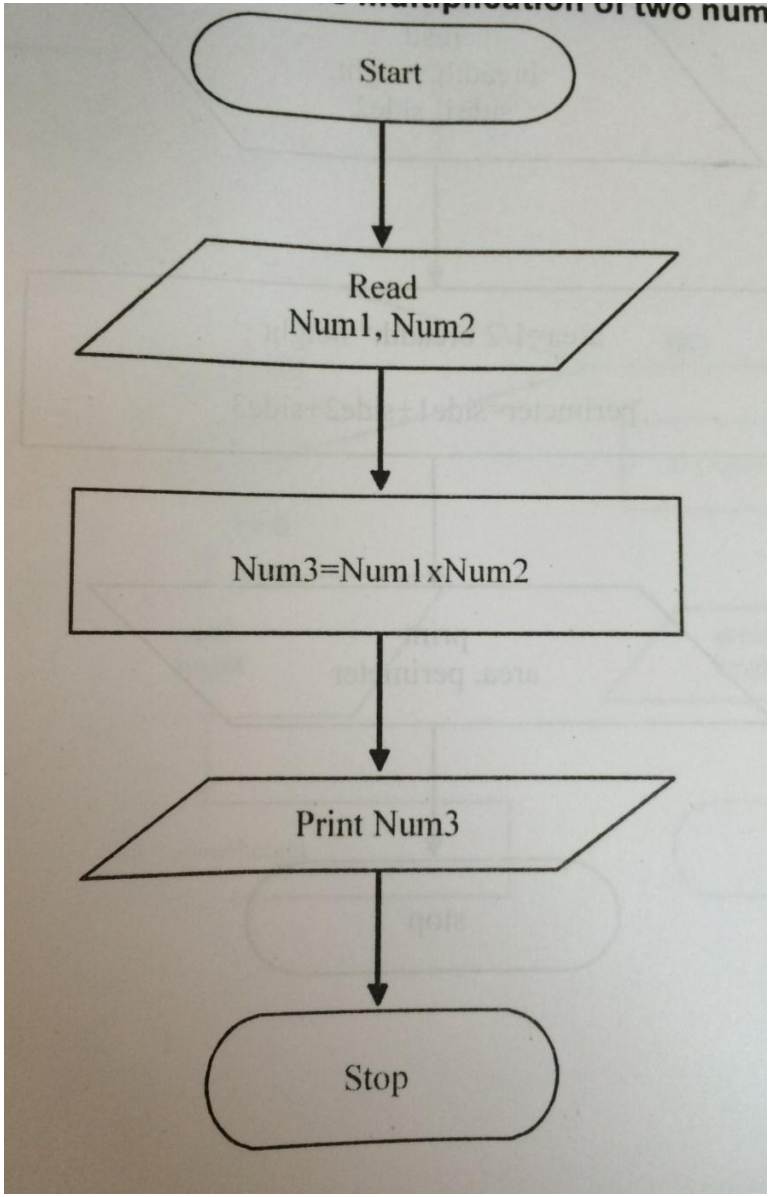

Start
read
stud1mark, stud2mark,
stud3mark
Is
stud1mark>
stud2mark?
YES
NO
Is
stud1mark>
stud3mark?
YES
NO
Is
stud2mark>
stud3mark?
YES
NO
print stud1
mark
is the highest.
print stud3
mark
is the highest
print stud2
mark
is the highest.
print stud3
mark
is the highest.
stop


start
read number
sum=0
Is number>0
Yes
No
rem=number%10
sum=sum+rem
number=number/10
print sum
stop


Start
read n
fact=1
term=1
Is term<=n
NO
YES
fact=fact?term
term=term+1
print fact
Stop

raw a flowchart t

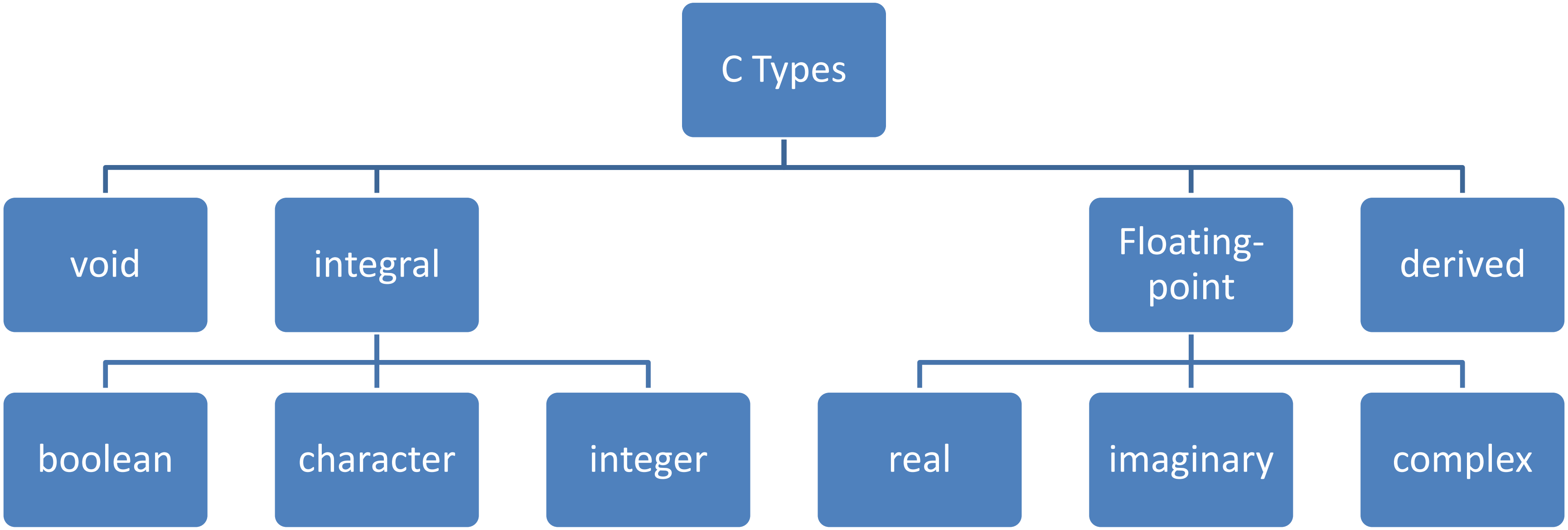



multiplication of two num
Start
Read
Num1, Num2
Num3=Num1xNum2
Print Num3
Stop

Exercise

- Explain whether following valid variable names and if not why?
- Sum + g_6
- _3_a_c
- 5_next_
- x++
- w.h.o.
- a$b
- in-change
- –beg.

- Explain the capacities and types of these variables:
- float f;
- char c;
- double d;
- int i;
- long int li;
- long double ld;



C Types

void

integral

Floating-point

derived

boolean

character

integer

real

imaginary

complex