

CPE 101: Programming the computer

Engr. Ms. ODIASE

Overview

- Humans cannot speak or write in computer language
 - Computers cannot speak or write human language
 - Therefore, an intermediate language is needed, known as programming language.
 - Programming languages allow a computer to direct the activities of the computer.
 - They are structured around a unique set of rules that dictate exactly how programmer should direct the computer to perform a specific task.
 - The rules of particular language tell the programmer how the individual instructions must be structured and what sequence of words and symbols must be used to form an instruction.
- ❖ **A computer instruction has two parts;**
1. Operation code
 2. Operand(s)

OPERATION CODE AND OPERANDS

- In computers, an operand is the part of a computer instruction that specifies data that is to be operated on or manipulated and, by extension, the data itself. Basically, a computer instruction describes an operation (add, subtract, and so forth) and the operand or operands on which the operation is to be performed
- The operation code tells the computer what to do such as add, subtract, multiply and divide.
- The operands tell the computer the data items involved in the operations. The operands in an instruction may consist of the actual data that the computer may use to perform an operation, or the storage address of data.
- Example: $a = b + 5$
- The “=” and “+” are operation codes while ‘a’, ‘b’ and 5 are operands.
- a and b can be storage addresses of actual data while 5 is an actual data.

Types of Instructions

- Input – output instructions
- Arithmetic instructions
- Branching instructions
- Logic instructions

❖ Instructions

- An input instruction directs the computer to accept data from a specific input device and store it in a specific location in the store.
- An output instruction tells the computer to move a piece of data from a computer storage location and record it on the output medium.
- All basic arithmetic operations can be performed by the computer. Since arithmetic operations involve at least two numbers, an arithmetic operation on the computer must include at least two operands.

❖ Branch Instructions

- Branch instructions cause the computer to alter the sequence of execution of instruction within the program.
- There are two types of branch instructions:
 1. Unconditional branch instruction / statement: cause the computer to branch to an instruction or statement regardless of the existing conditions.
 2. Conditional branch instruction/ statement : cause the computer to branch to a statement only when certain conditions exist.
- Logic instructions allow the computer to change the sequence of execution of instruction, depending on conditions, built into the program by the programmer. Typical logic operations include: shift, compare and test.

Types of Programming Language

- System software directs the internal operations of the computer, and applications software allows the programmer to use the computer to solve user made problems.
- Computer programming languages can be classified into the following:
 1. Machine language
 2. Assembly language
 3. High level language
 4. Very high level language

Machine Language

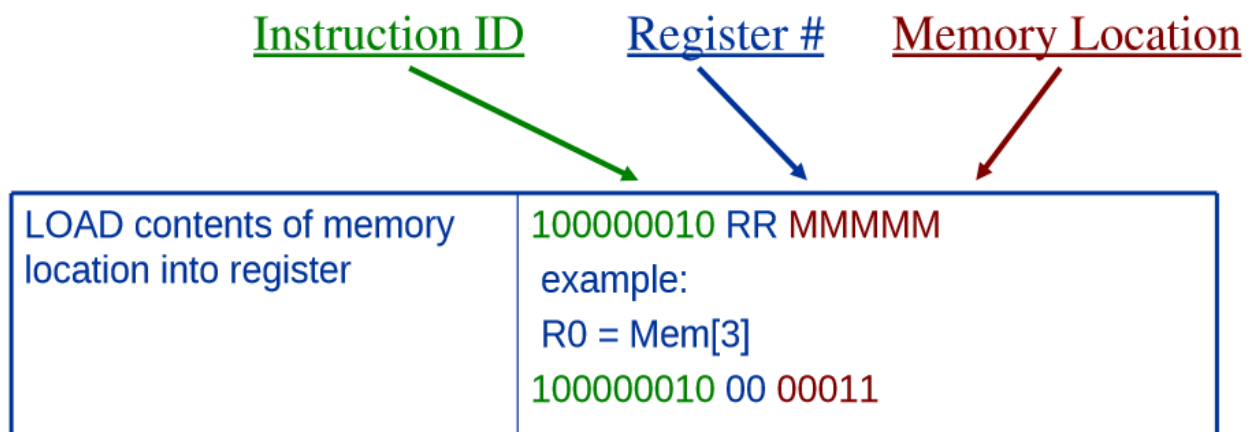
- Earliest form of programming language (First Generation Computer Language) structured in a binary number system.
- Programmers had to construct programs that used instructions written in binary notation, 1 and 0.
- Tedious to write
- Computer dependent:
The machine language for a particular computer is tied to the architecture of the CPU.
For example: G4 Macs have a different machine language than Intel PC's
- Time consuming
- Susceptible to errors
- Each instruction in a machine language program consists

of operation codes and operands.

- Operands of an instruction must tell the computer the storage locations of the data to be processed.

Machine language

Sample Instruction



- The programmer must deliberate storage locations for both instructions and data as part of the programming process.
- The programmer has to know the location of every switch and register that will be used in executing the program and must control their functions by means of instructions in the program.
- A machine language allows the programmer to take advantage of all features and capabilities of the computer

system for which it is designed.

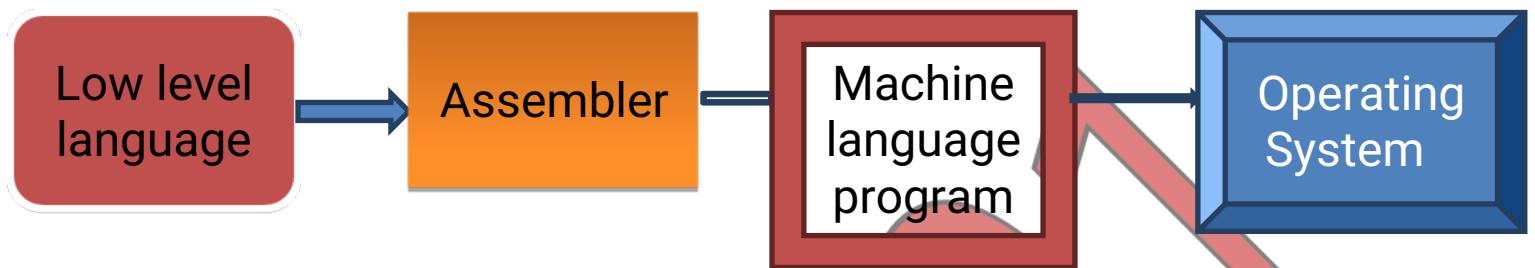
- Capable of producing the most efficient program as far as the storage requirements and operating speeds are concerned.

➤ **Assembly (Low level) Language**

- A symbolic way of expressing machine language was devised. In assembly language, the operation code is expressed as a combination of letters rather than binary numbers, sometimes called Mnemonics. Assembly language is a human-readable version of machine language.
- This allows the programmer to remember the operation code easily than when expressed strictly as binary numbers.
- Storage addresses / locations are expressed as a symbol rather than the actual numeric address.
- After the computer has read the program, operations software are used to establish the actual locations for each piece of data used by the program.
- Most popular is the IBM assembly language.
- The computer understands and executes machine language programs, the assembly language program must be translated into a machine language.
- This is accomplished by using a system software program called an Assembler. A tool called an assembler converts assembly language to machine language. It turns human readable list of instructions and directives into the encoded values the machine will execute. The assembled code is

called *object code*.

- The assembler accepts an assembly language program and produces a machine language program that the computer can actually execute.

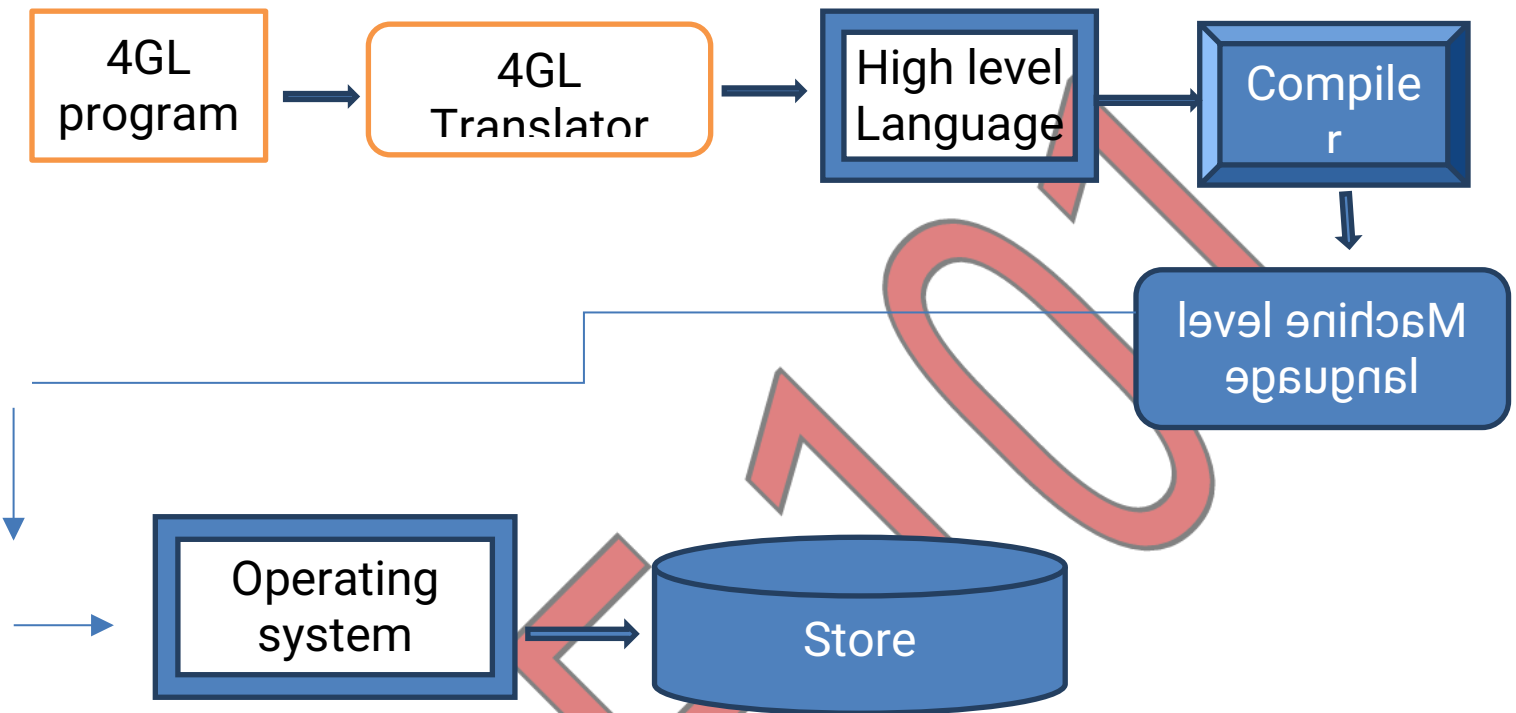


➤ Very High Level Language

- A fourth generation software intended to help computer users or computer programmers to develop their own application programs more quickly and cheaply.
- A 4GL by using a menu can allow users to specify what they require, rather than describe the procedures by which these requirements are met.
- Programming aids or tools are provided such as database management system, debuggers, program generators.
- Offers the user an English like set of commands and simple control structures in which to specify general data processing or numerical operations.
- Non procedural language
- Translated to a conventional high level language which is

passed to the compiler.

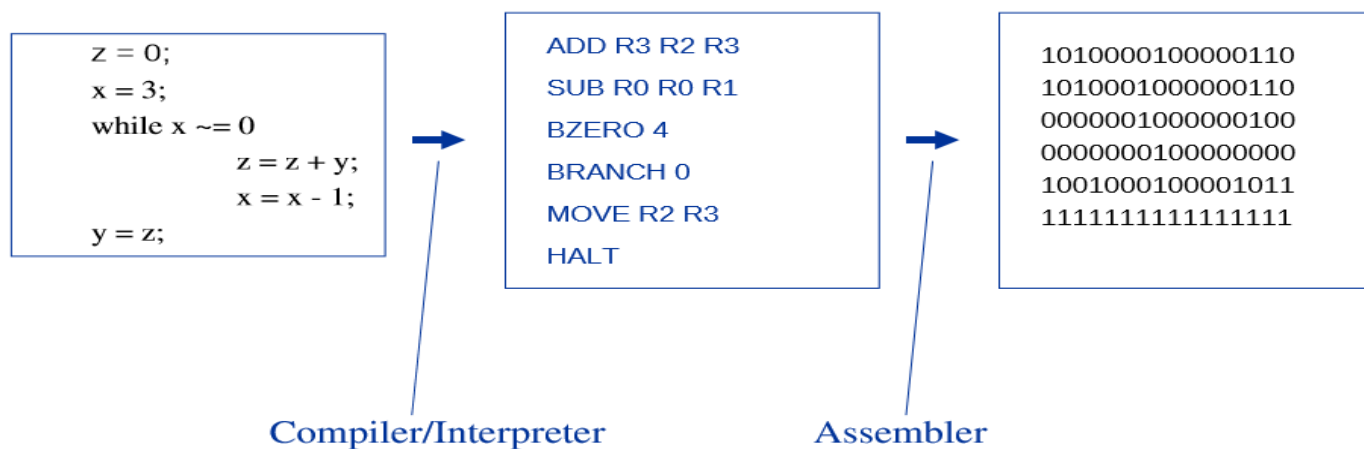
- Program flows are not designed by the programmer but the fourth generation software itself.
- Each user result is for a result rather than a procedure to obtain the result,



Compiler

- A **compiler** is a computer program that translates a program written in a high-level language to the machine language of a computer. The high-level program is referred to as 'the source code.' A typical computer program processes some type of input data to produce output data. The compiler is used to translate source code into machine code or compiled code. This does not yet use any of the input data. When the compiled code is executed, referred to as 'running the program,' the program processes the input data to

produce the desired output.



- Platform Independent: A program with instructions that can be executed without modification on several different types of hardware. Examples: Many HLL programs (C/C++, Pascal, COBOL, BASIC)
- Platform Dependent: A program with instructions that can only be executed on a particular type of hardware. Examples:
 - Assembly language programs

• Computer Programming

- Computer programming is the act of writing a program that a computer can execute to produce the desired result.
- A program is a series of instructions assembled to enable the computer to carry out a specified procedure.

- A computer program is a sequence of simple instructions into which a given problem is reduced and which is in a form the computer can understand either directly or after translation.

● Principles of good programming

- Accuracy: the program must do what it is supposed to do correctly and must meet its specification.
- Reliability: must always do what It is supposed to do and never crash
- Efficiency: optimal utilization of resources is essential. The program must use available storage space and other resources in a way that the computer speed is not wasted.
- Robustness: The program should cope with invalid data and not stop without an indication of the cause/source of an error.
- Usability: the program must be easy enough to use and be well documented
- Maintainability: The program must be easy to amend, having good structuring and documentation.
- Readability: The codes in a program should be well laid out and explained with comments

●

● Stages of Programming

■ Can be classified into eight major steps:

1. Problem definition
2. Devising the method of solution
3. Developing the method using suitable aids e.g. pseudo code or flowchart
4. Writing the instructions in a programming language
5. Transcribing instructions into machine sensible form
6. Debugging the program
7. Testing the program
8. Documenting all work done in producing a program

■ Problem Definition: The first stage requires a good understanding of the problem. The programmer needs to thoroughly understand what is required of a problem. A complete and precise unambiguous statement of the problem to be solved must be stated which will entail a detailed specification which lays down the input processes and output required.

■ Devising the method of solution: The second stage involved is spelling out the detailed algorithm. The use of computer to solve problems requires that a procedure or an algorithm be developed for the computer to follow in solving the problem.

■ Developing the method of solution: There are several methods for representing solutions to a problem. E.g.

pseudocode, flowchart, algorithm and decision tables.

- Writing the instructions in a programming language: After outlining the method of solving the problem, a proper understanding of the syntax of the programming language to be used is necessary in order to write the series of instructions required to get the problem solved.
- Translating the instructions in to machine sensible form: After the program is coded, it is converted into machine language. There are some specially written programs that translate user programs (source programs) into machine language (object code) called Translators. Compilers translate instructions that machines can execute at a go, interpreters accepts a program and execute it line by line.

● Debugging

- Program Debugging: Debugging is the process of locating and correcting errors. Three classes of errors:
- Syntax (Grammatical) errors: caused by coding mistakes (illegal use of a feature of the programming language). If a program contains syntax error, it will not pass compilation.
- Logic (Semantic) errors: caused by faulty logic in the design of the program. The program will work but not as intended. If a program contains only semantic errors, it means that it can pass compilation, but does not do what it meant to do.
- Execution (Run- time) error: works as intended but illegal input or other circumstances at run time makes the program stop.

Runtime errors happen during program run-time

- Two levels of debugging:

Desk checking or dry running: performed after the program has been coded. Its purpose is to locate and remove as many logical and clerical errors as possible. The program is then processed by a translator. The function of the translator is to convert the program statements into the binary code of the computer called the object code. During this translator process, the program statements are detected and the language translator generates a series of diagnostics referred to as the error message list. With this list in the hand of the programmer, the second level is reached. The error message list helps the programmer to find the cause of the errors and make necessary corrections.

Program Testing & Documentation

- The purpose of testing is to determine whether a program consistently produces correct or expected results. A program is normally tested by executing it with a given set of input data called test data for which correct results are known.
- A documentation of the program should be developed at every stage of the programming cycle.

Advantages of Program Documentation

- It provides all necessary information for anyone who comes in contact with the program.
- It helps the supervisor in determining the programs purpose, how long the program will be useful and future revisions that may be necessary
- It simplifies program maintenance
- It provides information as to the use of the program to those unfamiliar with It
- It provides operating instructions to the computer operator.

Algorithms and Flowcharts

- A typical programming task can be divided into two phases: n
1. Problem solving phase
 - produce an ordered sequence of steps that describe solution of problem
 - this sequence of steps is called an algorithm
 2. Implementation phase
 - implement the program in some programming language

Steps in Problem Solving

- First produce a general algorithm (one can use

pseudocode)

- Refine the algorithm successively to get step by step detailed algorithm that is very close to a computer language.
- Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English. Pseudo-code is an informal high level description of the operating principle of an algorithm. Pseudocode is an informal tool that you can use to plan out your algorithms. As you begin to write more complex code, it can be hard to keep an entire program in your head before coding it. Think of pseudocode as a step-by-step verbal outline of your code that you can later transcribe into a programming language. It is a combination of human language and programming language: it mimics the syntax of actual computer code, but it is more concerned with readability than with technical specificity.

Algorithm

- Algorithm: defined unit sequence of operations or activities which gives an unambiguous method of solving a problem or finding out that solution exists.
- A set of rules for carrying out calculations either by hand or using a machine
- A finite step by step procedure to achieve a required result
- A sequence of computational steps that transform the input into the output

- an abstraction of a program to be executed on a physical machine

✓ Quick Example

- ✓ Pseudocode & Algorithm

Example 1: Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Pseudocode:

Input a set of 4 marks

Calculate their average by summing and dividing by 4

if average is below 50

Print "FAIL"

else Print "PASS"

✓ Example

- Detailed Algorithm
- Step 1: Input M1,M2,M3,M4
- Step 2: $GRADE \leftarrow (M1+M2+M3+M4)/4$
- Step 3: if (GRADE < 50)
then Print "FAIL"




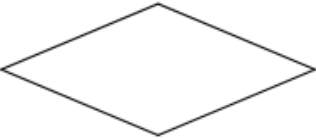



```
else Print "PASS"
```

```
endif
```

Flowchart

- A graphical representation of the major steps in solving a task.
- Displays in separate boxes (illustration symbols) the essential steps and shows by means of arrows the direction of information flow.
- Can also be defined as a graphical representation of an algorithm.
- A Flowchart
 - shows logic of an algorithm
 - emphasizes individual steps and their interconnections
 - e.g. control flow from one action to the next

Flow chart symbols

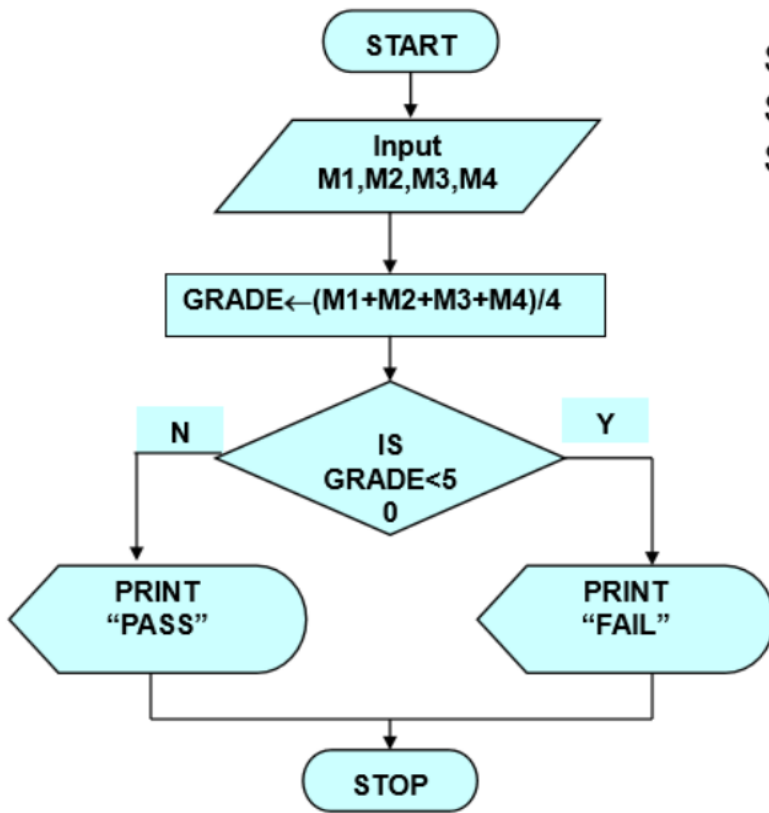
Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the program

❖ flowcharts

- Read from top to bottom
- Written notes inside each symbol to indicate specific functions
- Sequence of operations is performed until a terminal symbol designates the end of the run or branch connector transfers control

Flowchart

CPE101



Step 1: Input M1,M2,M3,M4
Step 2: $GRADE \leftarrow (M1 + M2 + M3 + M4) / 4$
Step 3: if (GRADE < 50) then
 Print "FAIL"
 else
 Print "PASS"
 endif

Examples of Flowcharts and Algorithms

- Write an algorithm to read values for three variables: X, Y and Z and find the value for RESULT from the formula:

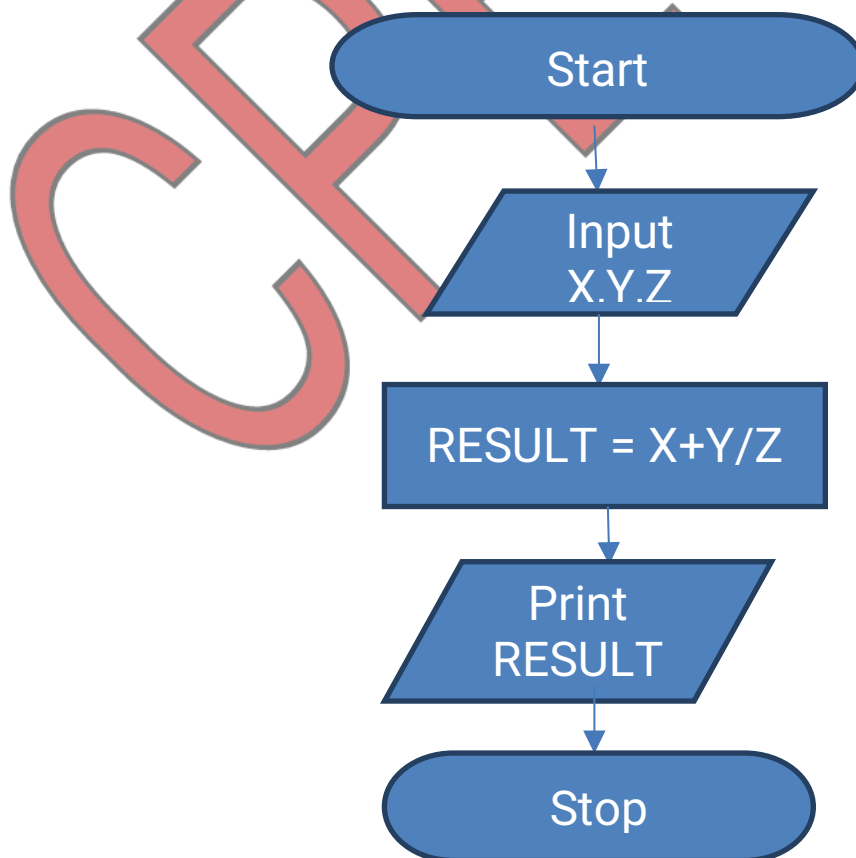
- $\text{Result} = X + Y/Z$

✓ Solution

- Algorithmic Solution

- I. Start
- II. Input values for X, Y, Z
- III. Compute value for result, $\text{result} = X + Y/Z$
- IV. Print value of result
- V. Stop

Flowchart



✓ *More examples*

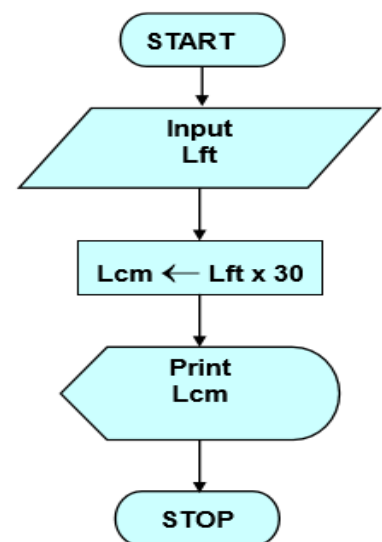
- *Write an algorithm and draw a flowchart to convert the length in feet to centimetre.*
- *Pseudocode:*
- *Input the length in feet (Lft)*
- *Calculate the length in cm (Lcm) by multiplying LFT with 30*
- *Print length in cm (LCM)*

cont'd

Algorithm

- Step 1: Input Lft
- Step 2: $Lcm \leftarrow Lft \times 30$
- Step 3: Print Lcm

Flowchart



- Class work
- Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Pseudocode

Input the width (W) and Length (L) of a rectangle

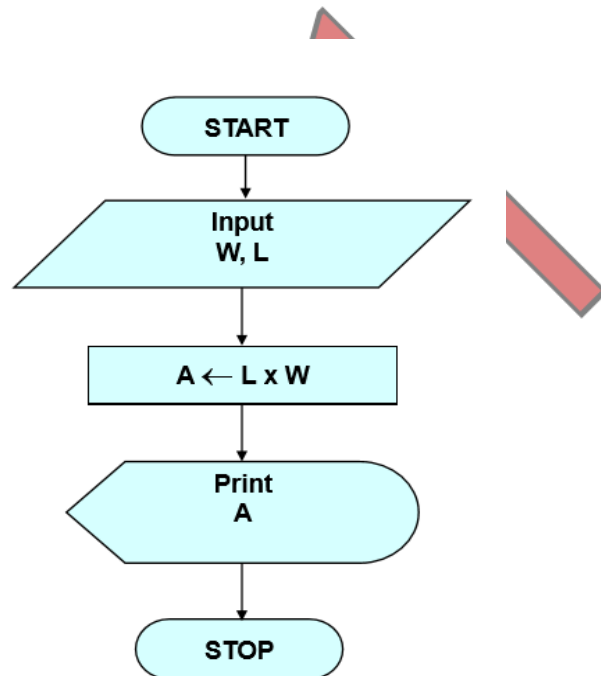
Calculate the area (A) by multiplying L with W

Print A

✓ Answers

Algorithm

- Step 1: Input W,L
- Step 2: $A \leftarrow L \times W$
- Step 3: Print A



CV