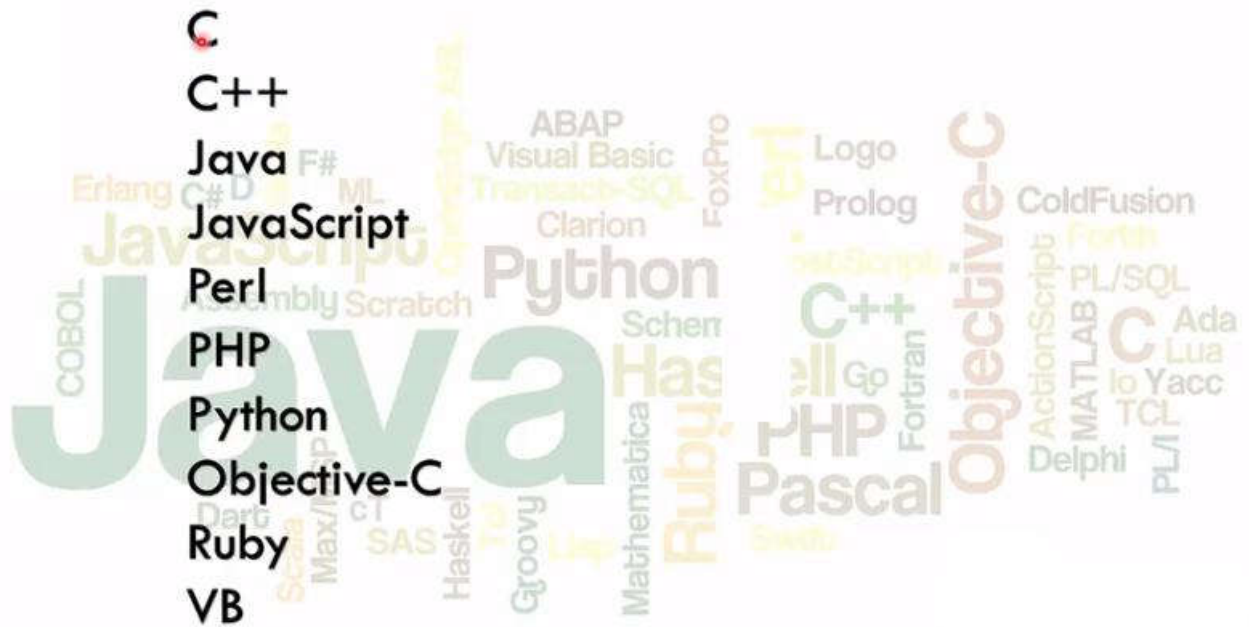
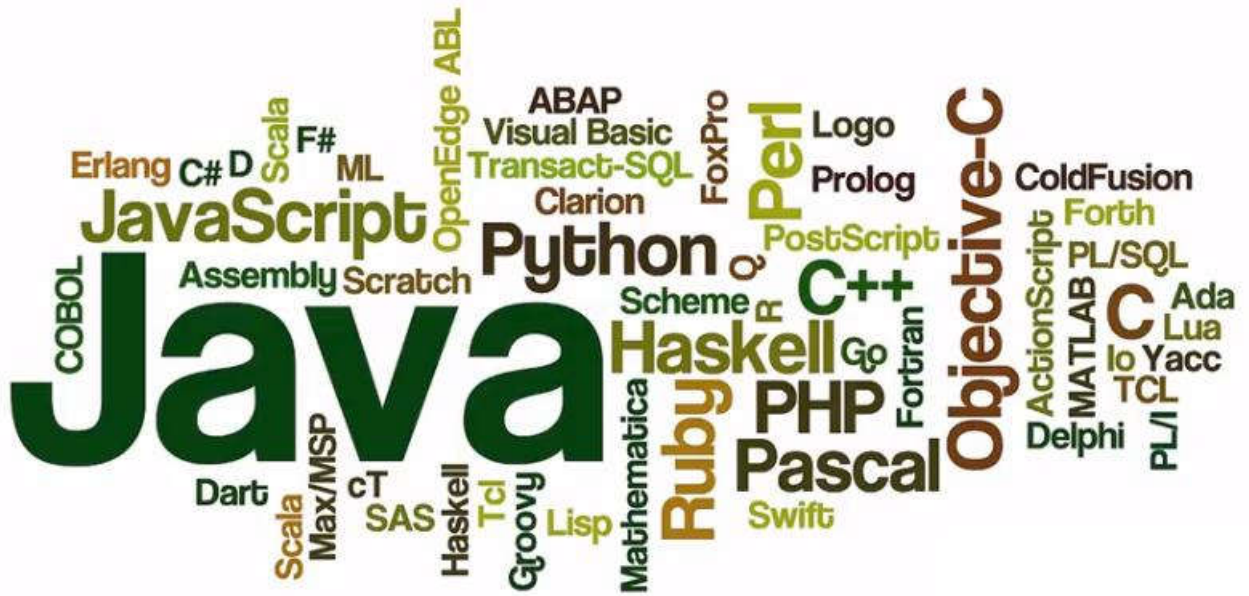
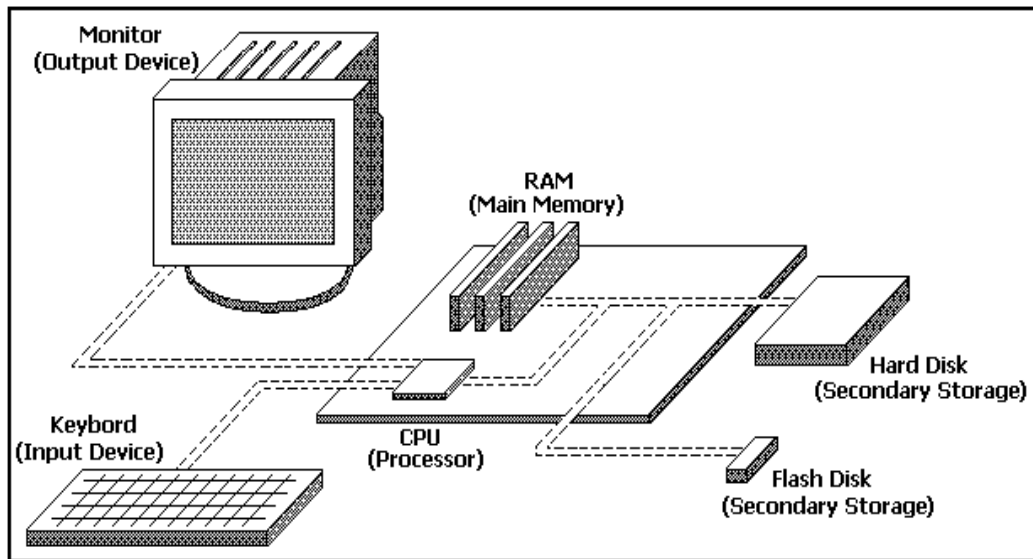


Computer Programming I

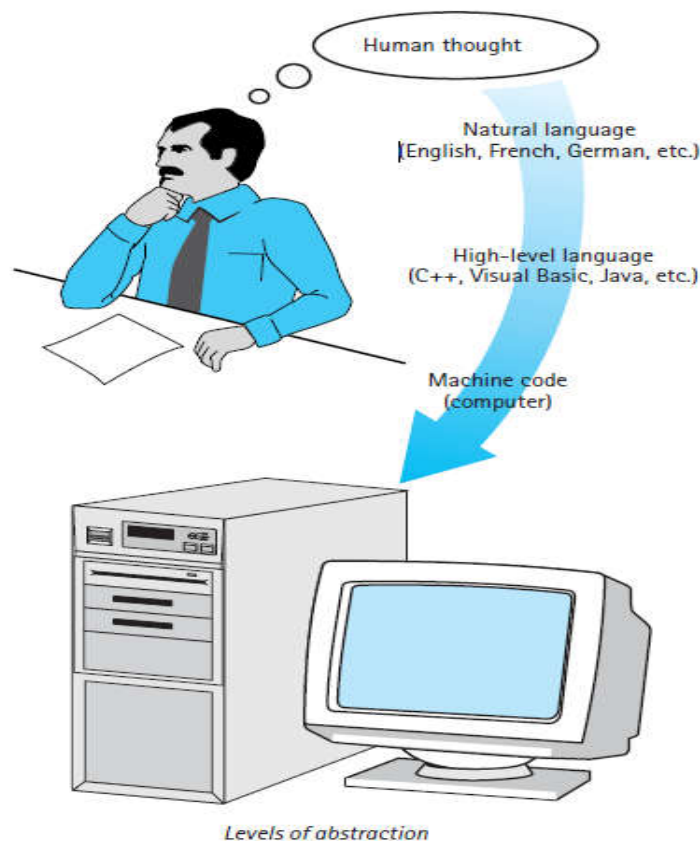


Introduction



- ✚ **Computer**: is a device capable of performing computations and making logical decisions at speeds millions and even billions of times faster than human beings.
- ✚ **Programming** is the process of writing instructions for a computer in a certain order to solve a problem.
- ✚ The computer programs that run on a computer are referred to as **software**. While the hard component of it is called **Hardware**.

Problem solving

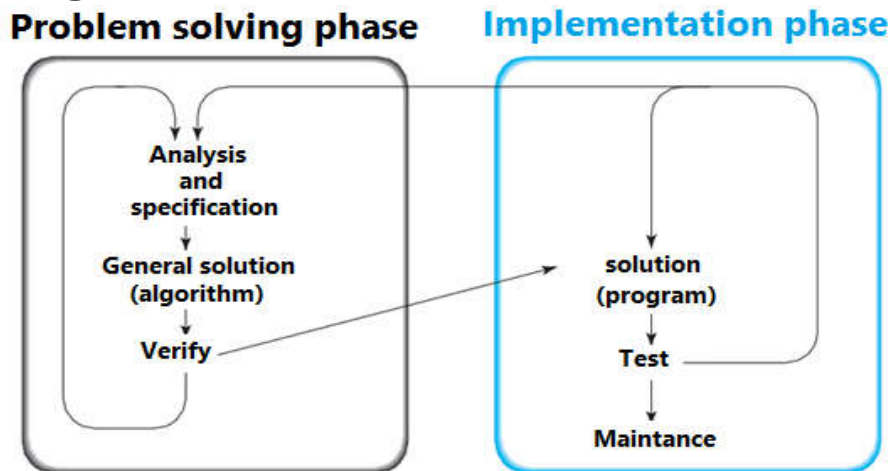


- **Problem:** A question raised for solution.
- **Solving:** finding a solution for something.

So, the problem solving is the act of finding a solution to problem

- The result of the **problem solving** is an algorithm, expressed in English.
- To produce a program in a programming language such as C++, the algorithm is translated into the programming language.

Program Design Process



Problem Solving Phase:

- ✚ **Analysis and Specification:** Understand (define) the problem and what the solution must do.
- ✚ **General Solution (Algorithm):** Specify the required data types and the logical sequences of steps that solve the problem.
- ✚ **Verify:** Follow the steps exactly to see if the solution really does solve the problem.

Implementation Phase:

- ✚ **Solution (Program):** Translate the algorithm into a programming language.
- ✚ **Test:** Manually check the results. If you find errors, analyze the program and the algorithm to determine the source of the errors, and then make corrections.
- ✚ **Maintenance phase:** Modify the program to meet changing requirements or to correct any errors that show up while using it.

Example: Find value of the variable output of the equation: $Z = (x-y)^2$

Analysis and Specification:

- 1- Understand the question: is the account (حساب) value of the variable Z, therefore must determine the inputs are x and y, and then finding exact value of the variable Z previous equation.
- 2- Analysis stage: a review of the different ways to resolve and choose the most suitable in terms of speed, ease and accuracy.

General solution:

First way: value of the variable Z is calculate equation $Z=(x-y)^2$

1. set value of each variable x and y
2. find output x-y
3. finding value of variable Z by square of step 2

Second way: value of the variable Z is calculate equation $Z = x^{**2} - 2 * x * y + y^{**2}$

1. Compensation value of each variable x and y
2. Find square of x ($x^{** 2}$)
3. Find value of $2 * x * y$
4. Find the square of y ($y^{** 2}$)
5. Find subtraction the result of step3 from step 2
6. Finding the value of Z by addition step5 value with step4 value

Analyzed the previous two methods it is clear that the first faster, easier and more accuracy to reach the solution.

Algorithms

What are algorithms

- Algorithm , It called that name in relation to the Muslim world **Abu Ja'far Muhammad ibn Musa al-Khwarizmi**
- In the science of algorithms no fixed rules algorithms to represent the algorithm in this way, but there are some controls (ضوابط) that must be taken into account (اعتبار بعين) during the representation, and are:
 - not matter (لا يهم) use of any type of human languages (Arabic, English, French, ...).
 - preferably used words as easy as possible and clear.
 - It must be consists of only three structures : sequence, choice, repetition
 - Stay away (ابتعد) from the use of words have meaning is limited to a specific programming language.

Algorithm Definition

An algorithm can be defined as a finite sequence of effect statements to solve a problem. An effective statement is a clear instruction that can be carried out (تنفيذه).

Algorithm Properties:

- **Finiteness:** The algorithm must terminate a finite numbers of steps.
- **Non-ambiguity:** Each step must be clearly defined.
- **Effectiveness:** The algorithm should solve the problem in a reasonable amount of time.
- **Algorithm Language:** not matter use of any type of human languages (Arabic, English, French, ...).

Why we need the Algorithms

- Documentation of thinking in order to solve problems code.
- Determine the time and storage space that the computer needs to resolve the problem.
- Contribute (المساهمة) to speed the discovery of errors before you start thinking in practical application stage.
- Give us the opportunity (فرصة) to solve problems in different ways.

Algorithm Representation Ways

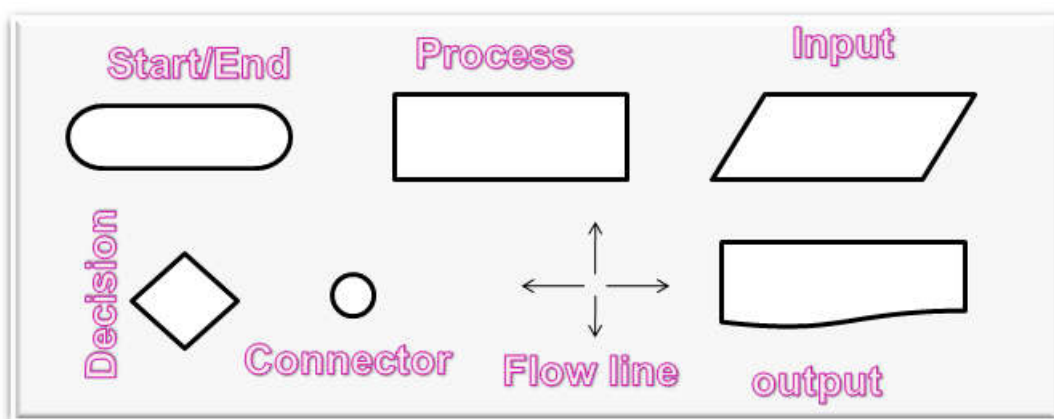
The Algorithm Representation in many ways like:

- Natural language
- Pseudo code
- Flow Chart

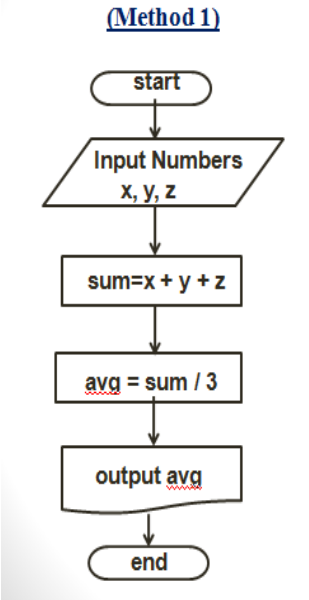
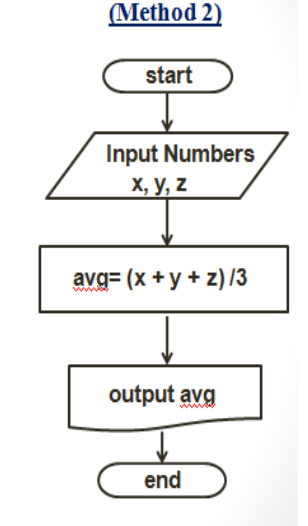
Natural language	Pseudo code	Flow Chart
1. Is way directly to express the solution by sentences and phrases natural languages: English , Arabic, .. 2. Differ from person to person	1. A clever way to represent algorithm 2. Similar to human language not considered a programming language. 3. Easily converted for different programming languages.	1. Symbolic representation to the algorithm. 2. Do not need to express your own language 3. It a lot easier

Flowcharts

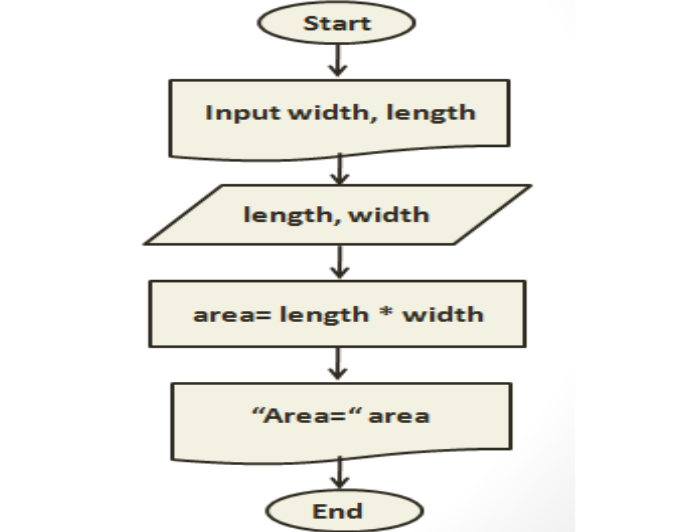
A flowchart is a graphical representation of an algorithm. Flowcharts are drawn using symbols. The main symbols used to draw a flowchart are shown in the following:



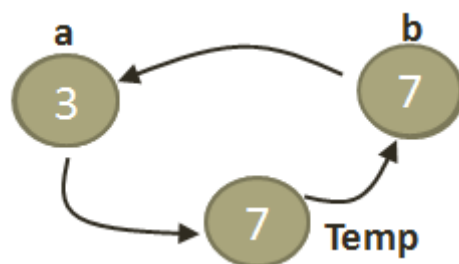
Example 1: Write an algorithm that Find the average of the three numbers.

By Natural language	By Pseudo code	By Flow Chart	
<ol style="list-style-type: none"> 1. Start. 2. Read the three numbers. 3. Calculate the sum of the three numbers. 4. Calculate the average by divide the sum by three. 5. Output the average 6. End. 	<p>Method1</p> <ol style="list-style-type: none"> 1. start. 2. input x, y and z. 3. $sum = x + y + z$. 4. $avg = sum / 3$. 5. output avg. 6. end. <p>Method2</p> <ol style="list-style-type: none"> 1. start. 2. input x, y and z. 3. $avg = (x + y + z) / 3$. 4. output avg. 5. end. 	<p>(Method 1)</p> 	<p>(Method 2)</p> 

Example 2: Write an algorithm that outputs the rectangle area given width and length.

By Pseudo code	By Flow Chart
<ol style="list-style-type: none"> 1. Start 2. output " Input width and length -> " 3. input length ,width 4. $area = length * width$ 5. output "Area = " , area 6. end 	

Example 3: Write an algorithm that can swapping between two inputs variables.



By Pseudo code	By Flow Chart
<ol style="list-style-type: none"> 1. start 2. output " input two numbers :- " 3. input a , b 4. temp = a 5. a = b 6. b = temp 7. output " After swapping " 8. output "a= ", a, "b= ", b 9. end 	<pre> graph TD Start([Start]) --> Input[Input two numbers] Input --> Read[/a, b/] Read --> Process[temp = a a = b b = temp] Process --> Output[/After swapping "a= "a, "b= " , b/] Output --> End([End]) </pre>

Control Structure

Any algorithm can be written using only three structures (together, individual), **Sequence, choice, repetition**, the following table describe thee representation of these structure by flowchart .

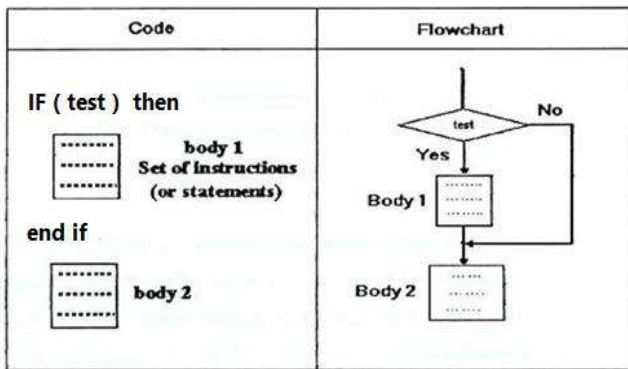
Sequence

Code	Flowchart
<pre> statement 1 statement 2 statement n </pre>	<pre> graph TD S1[statement 1] --> S2[statement 2] S2 --> Dots[...] Dots --> Sn[statement n] </pre>

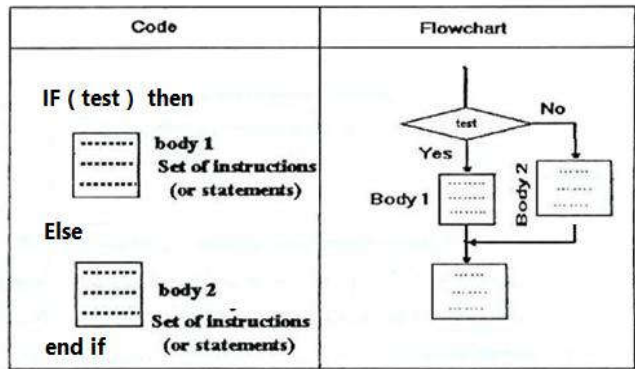
While loop structure

Code	Flowchart
<pre> while (test) do body Set of Instructions (or statements) end while </pre>	<pre> graph TD Test{test} -- Yes --> Body[Body] Body --> Test Test -- No --> Exit[...] </pre>

IF structure



IF - Else structure



Example 4: Write algorithm to print “pass” if student grade greater than 60 otherwise prints “fail”

By Natural language	By Pseudo code	By Flow Chart
<ol style="list-style-type: none"> start input student grade If students grade is greater than 60 Output "passed" else output "failed" end 	<ol style="list-style-type: none"> start input grade if grade > 60 then Output "passed" Else Output "failed" end 	

Example 5: Write an algorithm that can find the large number between two inputs numbers.

Answer:

By Pseudo code	By Flow Chart
<ol style="list-style-type: none"> start output "input two numbers :- " input a , b max = a if b > a then max=b output " Max= ", max end 	

Example 6: Write an algorithm can find summation of N numbers.

Answer: By Pseudo code

1. start
2. output " Number N :- "
3. input n
4. counter = 0 , sum= 0
5. if counter >= n then goto Step 11
6. counter = counter + 1
7. output " X= "
8. input x
9. sum = sum + x
10. goto Step5
11. output "Sum= ", sum
12. end

by flowchart

