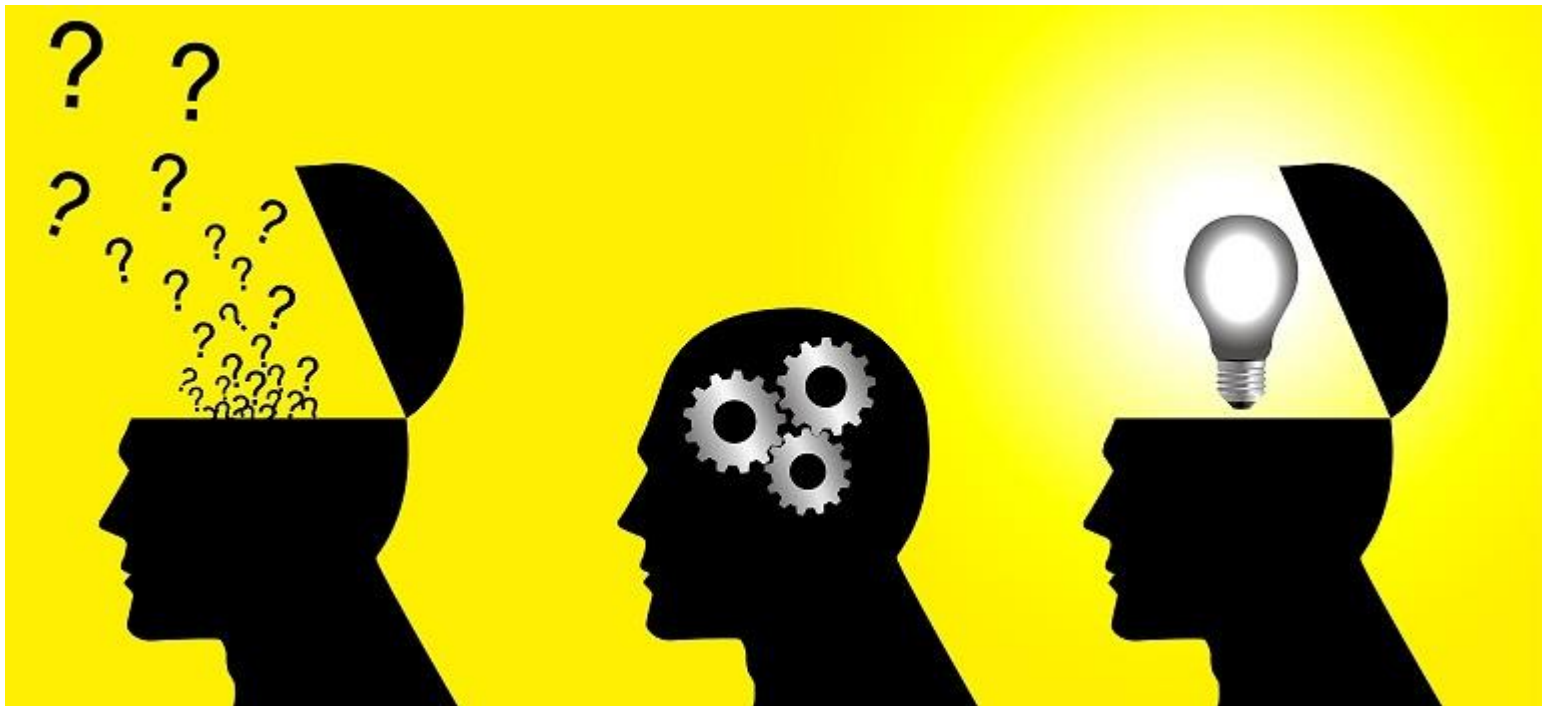


Programming Intro

What is Computer Science?

- Computer science is fundamentally about computational problem solving

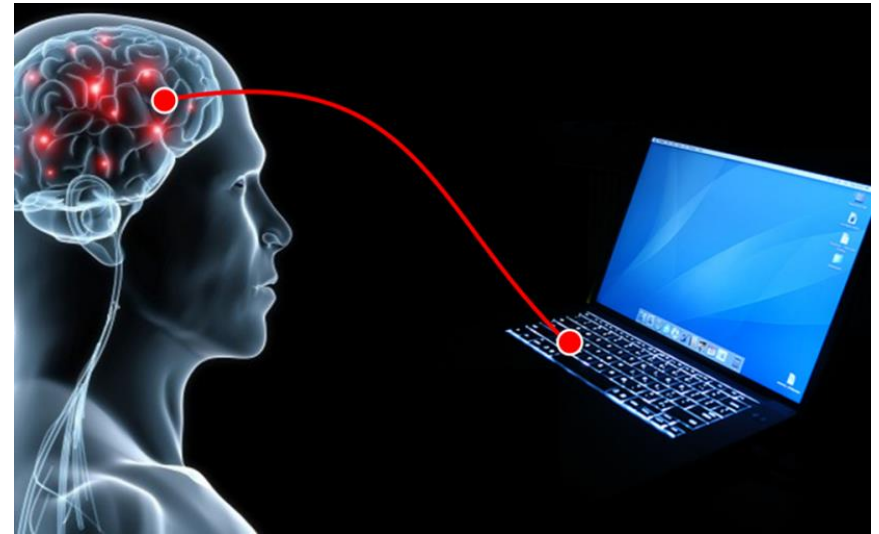


For instance: How lock-in patients can communicate?

“Locked-In” Syndrome

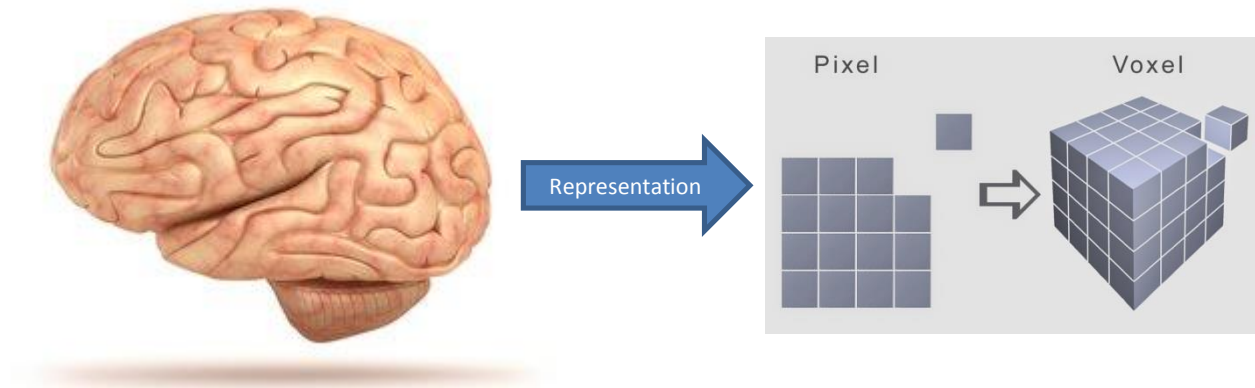
Plum & Posner (1966)

- **Quadriplegia**
 - Paralysis of Limbs
 - Anarthria
 - Loss of Articulate Speech
 - Aphonia
 - Loss of Vocalization
- **Full Consciousness**
 - Preserved Auditory, Visual Function
 - Startle, Orienting
 - Localization, Fixation, Pursuit
 - Preserved Communication
 - Blinking, Vertical Eye Movements
 - Preserved Emotional Response
- **Anterior Brain Stem**
 - Pons
 - Excludes Reticular Formation



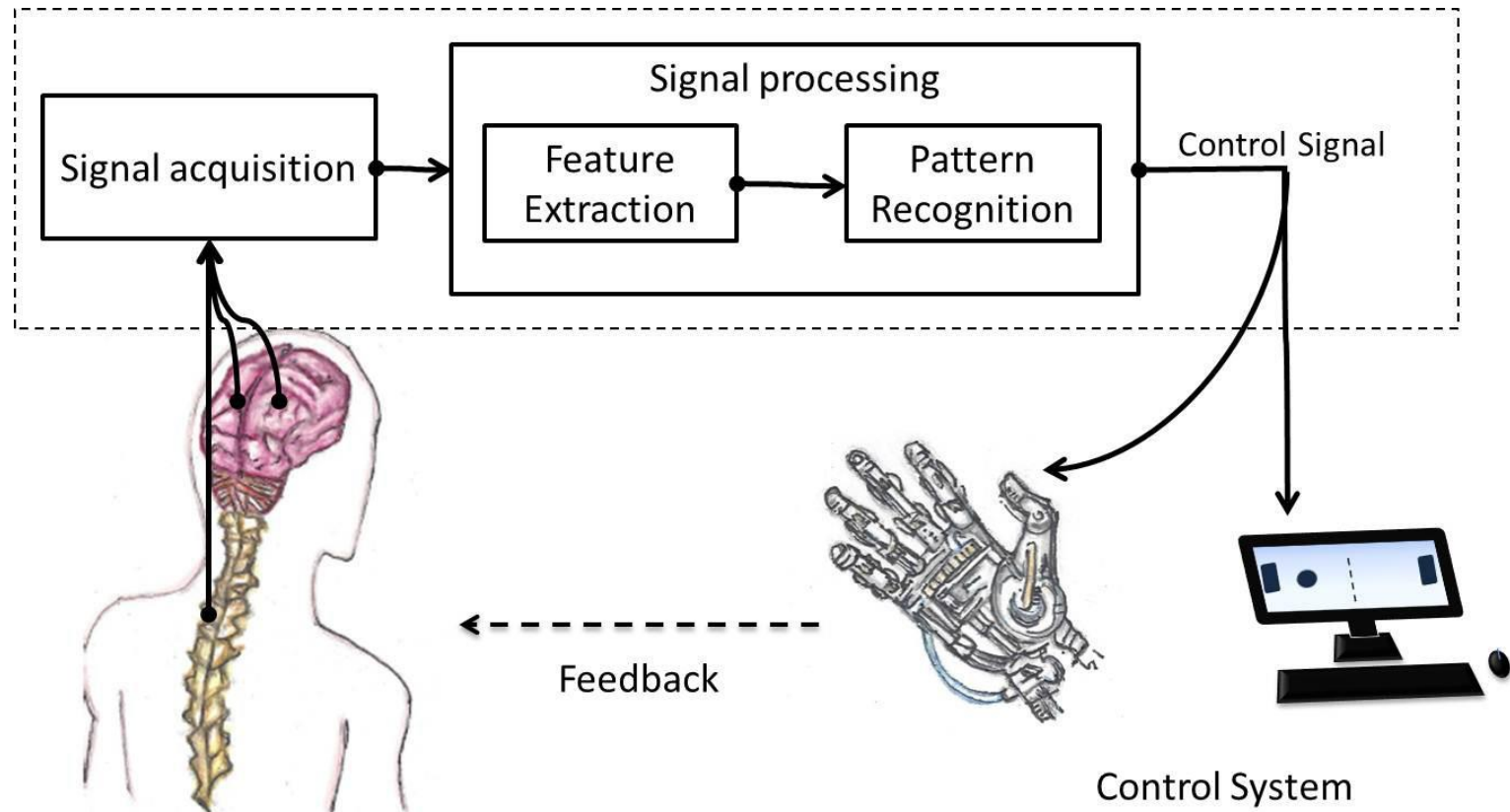
What we need to solve problems?

1. Representation. Captures all relevant aspects of a problem.



What we need to solve problems?

2. Algorithm. Recipe to solve the problem.

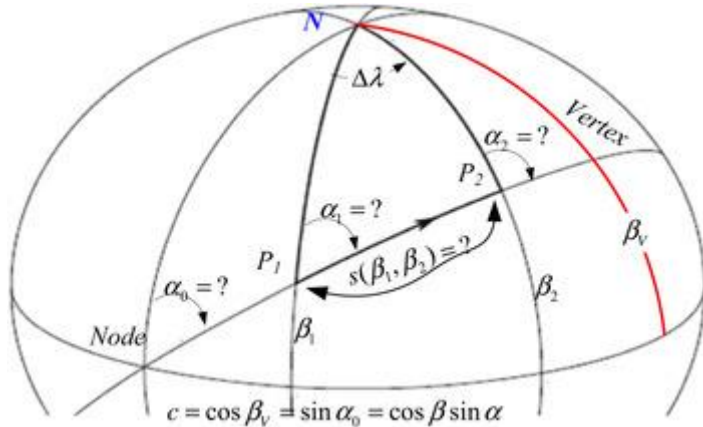


Algorithms

An **algorithm** is a finite number of clearly described, unambiguous “doable” steps that can be systematically followed to produce a desired result for given input in a finite amount of time.



Different kind of algorithms



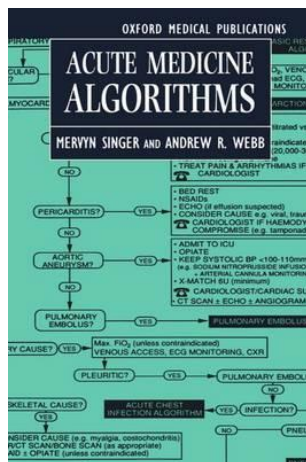
Vincent Formula. A fast algorithm to calculate the distance between two latitude/longitude points on an ellipsoid

EUCLID'S DIVISION ALGORITHM

• To obtain the HCF of two positive integer ,say 'c' and 'd' , with $c > d$, Follow the steps

1. Apply the division lemma to find 'q' and 'r' where $a = bq + r$, $0 < r < b$
2. If $r = 0$,, the HCF is b, if r is not equal to '0' apply Euclid lemma to 'd' and 'r'
3. Continue the process till the remainder is 'zero' . The division at this stage will be the required HCF

Euclid division algorithm



Diagnostic Algorithms

The Doomsday Algorithm - Calculating the Weekday of any given Date

In 1970, British mathematician John Conway devised a way to quickly calculate the weekday of any given date without the help of calculators, computers, or calendars

The best thing about the doomsday algorithm: your friends will think you have a superhuman memory, when all you need to do is memorize a set of numbers and do a series of simple calculations.

Conway's algorithm bases on the fact that some dates always fall on the same weekday within any given year. These dates are called *doomsdays*.

[Find out which dates are doomsdays ►](#)

[Try our new Doomsday Calculator ►](#)



Calculate the weekday of any date.

@Stockphoto.com/MariaPavlova

Day of the week

Dooms Da

1. The Do
2. The Do

[Find Next \(](#)

Friday

Create Cal

Full year

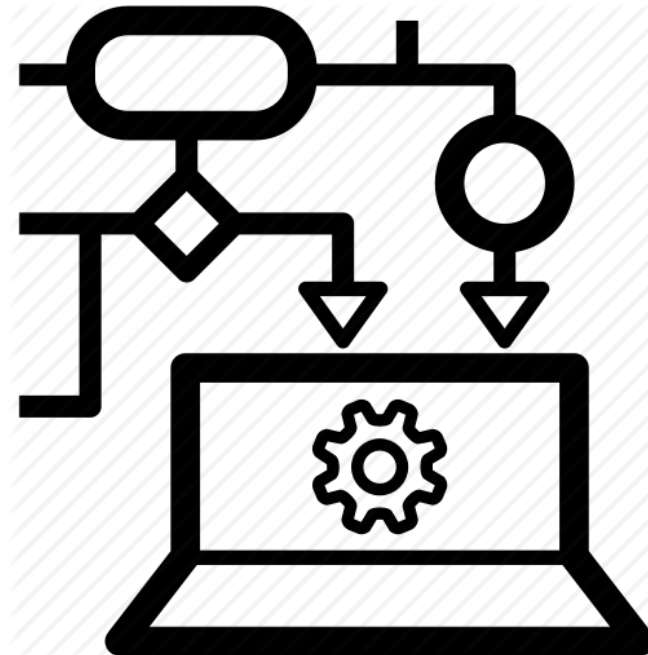
Cool algorithms



Algorithms are not just the playthings of lab rats. Many of them play a significant role in your daily life from helping to predict the weather to determining whether or not you ran that stop light on the way to work today. We decided to round up a few of the more interesting algorithms and look at how they impact your community.

Algorithms and Computers

- Computers can execute instructions very quickly and reliably without error, algorithms and computers are a **perfect match**.



Algorithm features

- **Input.** There are zero or more quantities that are externally provided.
- **Output.** At least one output is produced.
- **Definiteness.** Each instruction is clear and unambiguous.
- **Finiteness.** If we trace out the instruction of the algorithm, then for all cases, the algorithm terminates after a finite number of steps.
- **Effectiveness.** Every instruction must be basic enough to be carried out, in principle, by a person using only pencil and paper. Each operation must be feasible.

Examples of algorithms

- **Write an algorithm to add two numbers entered by user.**

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

$\text{sum} \leftarrow \text{num1} + \text{num2}$

Step 5: Display sum

Step 6: Stop

Examples of algorithms

- **Write an algorithm to find the largest among three different numbers entered by user.**

Step 1: Start

Step 2: Declare variables a, b and c.

Step 3: Read variables a, b and c.

Step 4: If $a > b$

 If $a > c$

 Display a is the largest number.

 Else

 Display c is the largest number.

Else

 If $b > c$

 Display b is the largest number.

 Else

 Display c is the greatest number.

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.**

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

$$D \leftarrow b^2 - 4ac$$

Step 4: If $D \geq 0$

$$r1 \leftarrow (-b + \sqrt{D})/2a$$

$$r2 \leftarrow (-b - \sqrt{D})/2a$$

Display r1 and r2 as roots.

Else

Calculate real part and imaginary part

$$rp \leftarrow -b/2a$$

$$ip \leftarrow \sqrt{(-D)/2a}$$

Display $rp + j(ip)$ and $rp - j(ip)$ as roots

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.**

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

$$D \leftarrow b^2 - 4ac$$

Step 4: If $D \geq 0$

$$r1 \leftarrow (-b + \sqrt{D}) / 2a$$

$$r2 \leftarrow (-b - \sqrt{D}) / 2a$$

Display r1 and r2 as roots.

Else

Calculate real part and imaginary part

$$rp \leftarrow -b / 2a$$

$$ip \leftarrow \sqrt{(-D) / 2a}$$

Display $rp + j(ip)$ and $rp - j(ip)$ as roots

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find the factorial of a number entered by user.**

Step 1: Start

Step 2: Declare variables n, factorial and i.

Step 3: Initialize variables

factorial \leftarrow 1

i \leftarrow 1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

5.1: factorial \leftarrow factorial*i

5.2: i \leftarrow i+1

Step 6: Display factorial

Step 7: Stop

Examples of algorithms

- **Write an algorithm to find the factorial of a number entered by user.**

Step 1: Start

Step 2: Declare variables n, factorial and i.

Step 3: Initialize variables

factorial \leftarrow 1

i \leftarrow 1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

5.1: factorial \leftarrow factorial*i

5.2: i \leftarrow i+1

Step 6: Display factorial

Step 7: Stop

Examples of algorithms

- **Write an algorithm to find the Fibonacci series till term ≤ 1000 .**

Step 1: Start

Step 2: Declare variables first_term, second_term and temp.

Step 3: Initialize variables first_term $\leftarrow 0$ second_term $\leftarrow 1$

Step 4: Display first_term and second_term

Step 5: Repeat the steps until second_term ≤ 1000

5.1: temp \leftarrow second_term

5.2: second_term \leftarrow second_term + first term

5.3: first_term \leftarrow temp

5.4: Display second_term

Step 6: Stop

Analysis of algorithms

*The theoretical study of **computer-program performance and resource usage.***

What's more important than performance?

- modularity
- correctness
- maintainability
- functionality
- robustness
- user-friendliness
- programmer time
- simplicity
- extensibility
- reliability









Analysis of algorithms

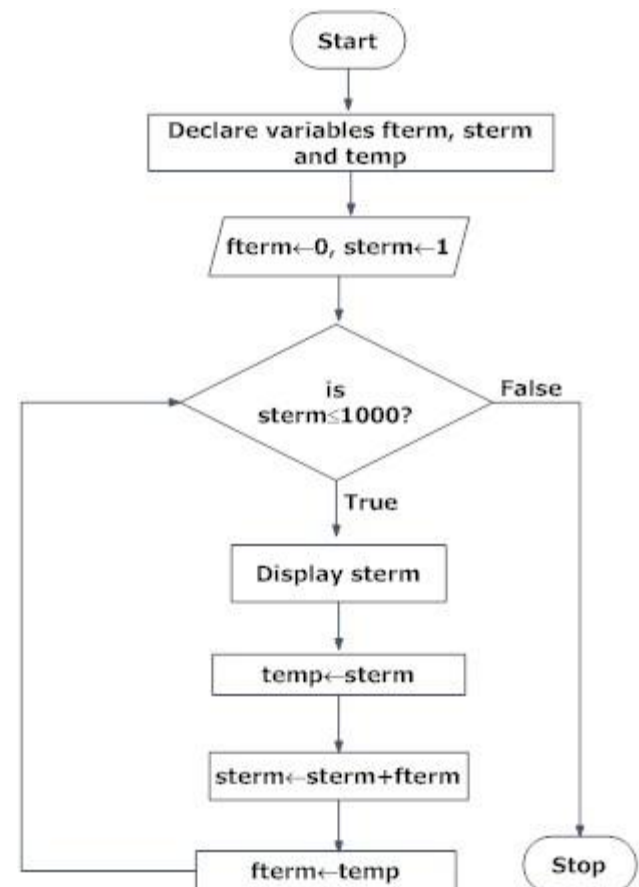
Why study algorithms and performance?

- Algorithms help us to understand **scalability**.
- Performance often draws the line between what is feasible and what is impossible.
- Algorithmic mathematics provides a **language** for talking about program behavior.
- The lessons of program performance generalize to other computing resources.
- Speed is fun!

Algorithms is about communication

Flow chart is a pictorial representation; algorithm is done through step by step direction.

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for airthmetic operations and data-manipulations.
	Desicion	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.



Algorithms and programs

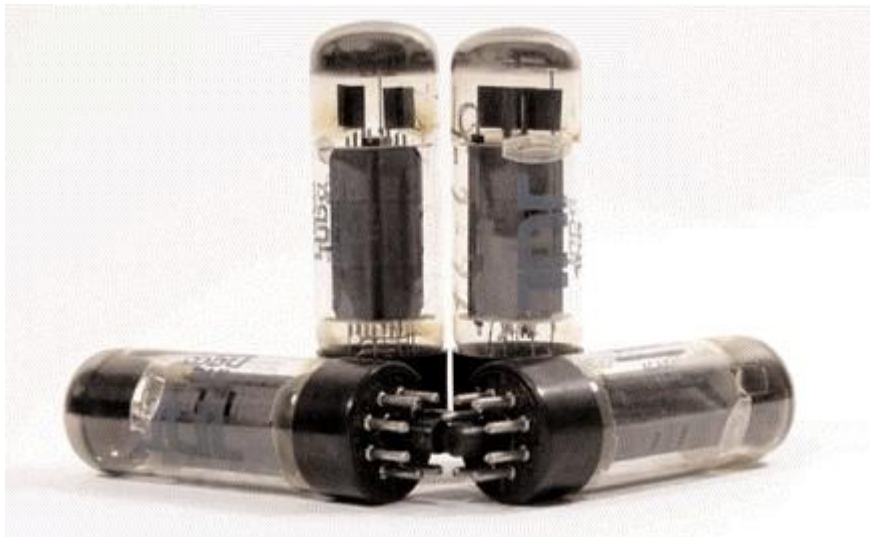
Algorithm is not the computer code. Algorithm are just the instructions which gives clear idea to you idea to write the computer code.



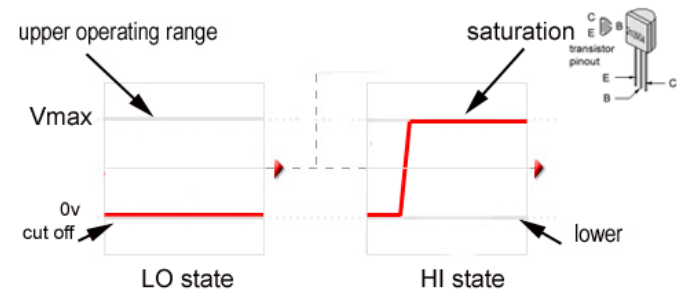
Program is a set of **computer instructions** written in a programming language

Computer Hardware

- All information within a digital computer system is represented using only two digits, 0 and 1, called **binary representation**.



Operating Range : transistors
have an operating range they work within



The 2 digital states Hi and Lo represent the transistor in 2 states off, and fully on (saturation). Analogue electronics operates on an infinity of voltages.

Binary System

place value in the binary system is based on 2

2^5	2^4	2^3	2^2	2^1	2^0
32's	16's	8's	4's	2's	1's
thirty-twos	sixteens	eights	fours	twos	ones
		1	1	0	0

= 12

a Base-2 system






© Jenny Eather 2014

The term **bit** stands for binary digit. A **byte** is a group of bits operated on as a single unit in a computer system, usually consisting of eight bits.

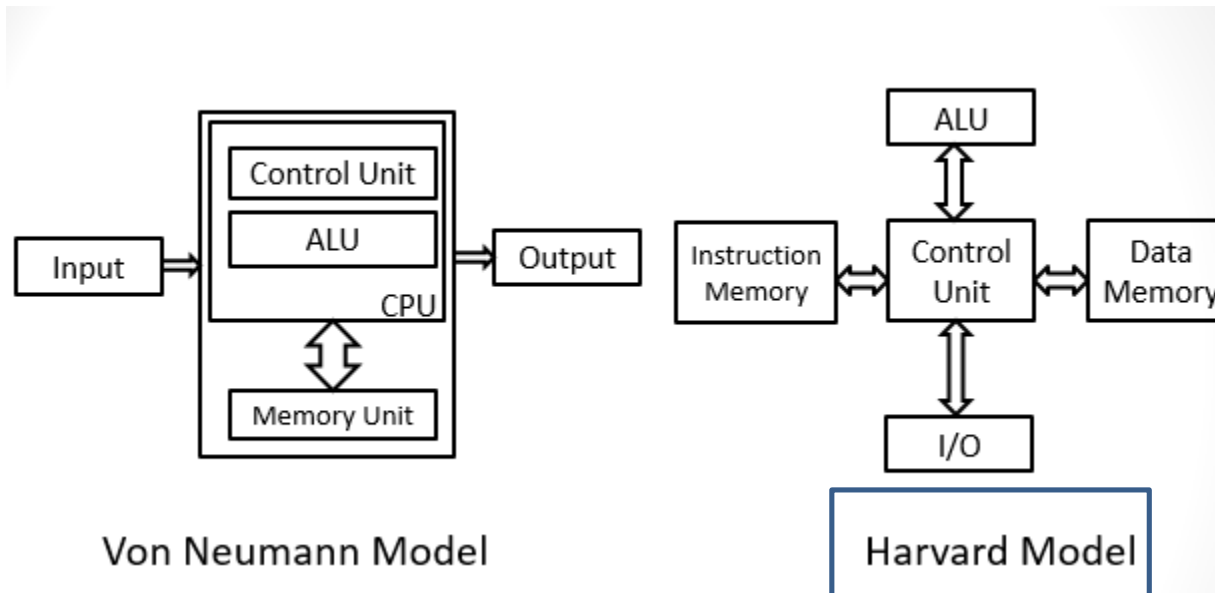
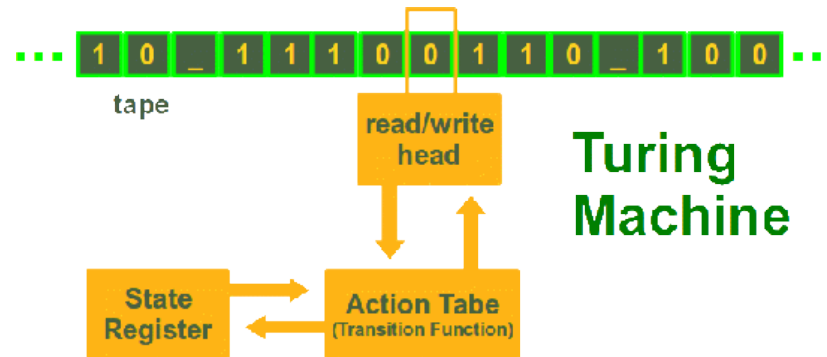
Hay 4 combinaciones
posibles con dos *bits*

Bit 1	Bit 0
 0	 0
 0	 1
 1	 0
 1	 1

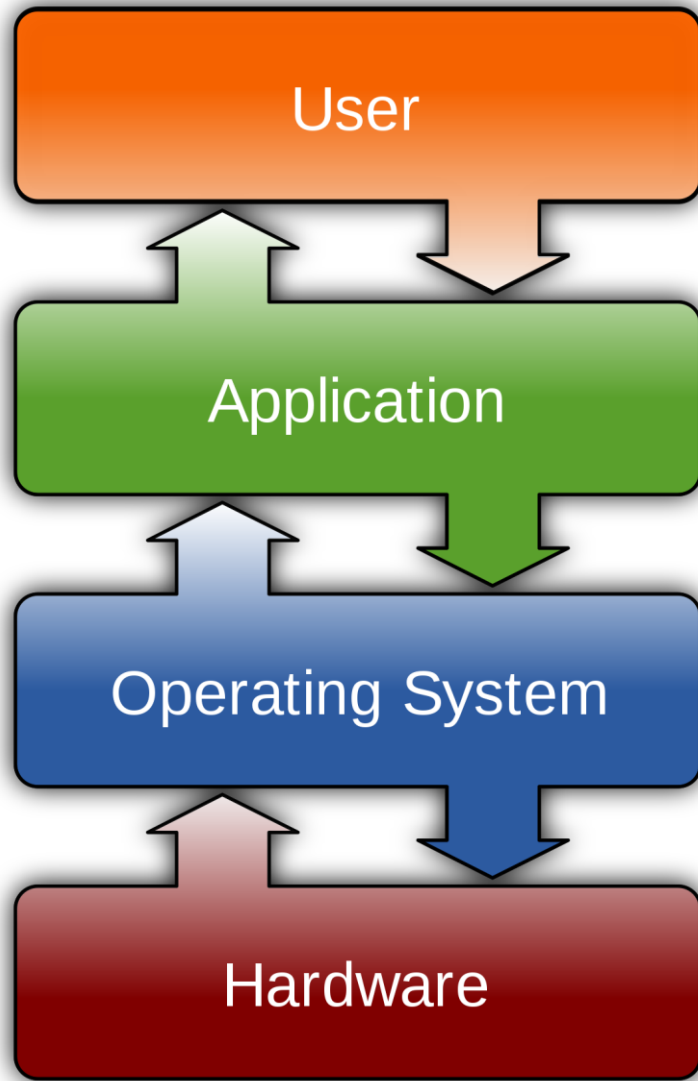
$$4 + 1 + 0,25 = 5,25$$

4	2	1	1/2	1/4	← Valor de posición
					Representación gráfica de los <i>bits</i> como bombillas encendidas y apagadas
1	0	1	0	1	
					← Dígitos binarios (<i>bits</i>)

Computer Architecture



Operative system



Teach-ICT.com




Computer Software

Computer software is a set of program instructions, including related data and documentation, that can be executed by computer.

ADA LOVELACE


FIRST COMPUTER PROGRAMMER




The Analytical Engine

Lovelace's program turned a complex formula into simple calculations that could be encoded on punched cards and fed into Charles Babbage's Analytical Engine, a mechanical computer that he designed but never built. She published it in 1843, a century before the modern computer age.

"I want to put in something about Bernoulli's Number, in one of my Notes, as an example of how an explicit function may be worked out by the engine, without having been worked out by human head and hands first."



$$\frac{x}{e^x - 1} = \frac{1}{1 + \frac{x}{2} + \frac{x^2}{2 \cdot 3} + \frac{x^3}{2 \cdot 3 \cdot 4} + \&c.}$$




A Universal Computer


Lovelace did more than write the first computer program. She was also the first person to realise that a general purpose computer could do anything, given the right data and instructions.

"The Analytical Engine weaves algebraic patterns just as the Jacquard loom weaves flowers and leaves."

"Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."



**Augusta Ada King,
Countess of Lovelace**
Born: 10 December 1815
Died: 27 November 1852



Ada Lovelace Day
FindingAda.com

The first program



How do we write computer instructions?

Step 1: Start

Step 2: Declare variables

first_term, second_term and temp.

Step 3: Initialize variables $\text{first_term} \leftarrow 0$

$\text{second_term} \leftarrow 1$

Step 4: Display first_term and second_term

Step 5: Repeat the steps until

$\text{second_term} \leq 1000$

5.1: $\text{temp} \leftarrow \text{second_term}$

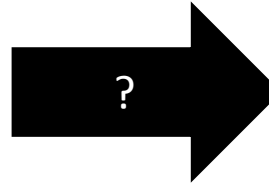
5.2: $\text{second_term} \leftarrow \text{second_term} + \text{first_term}$

term

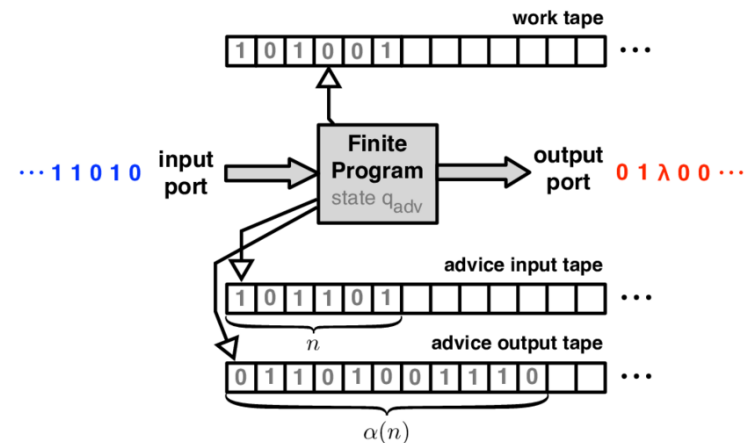
5.3: $\text{first_term} \leftarrow \text{temp}$

5.4: Display second_term

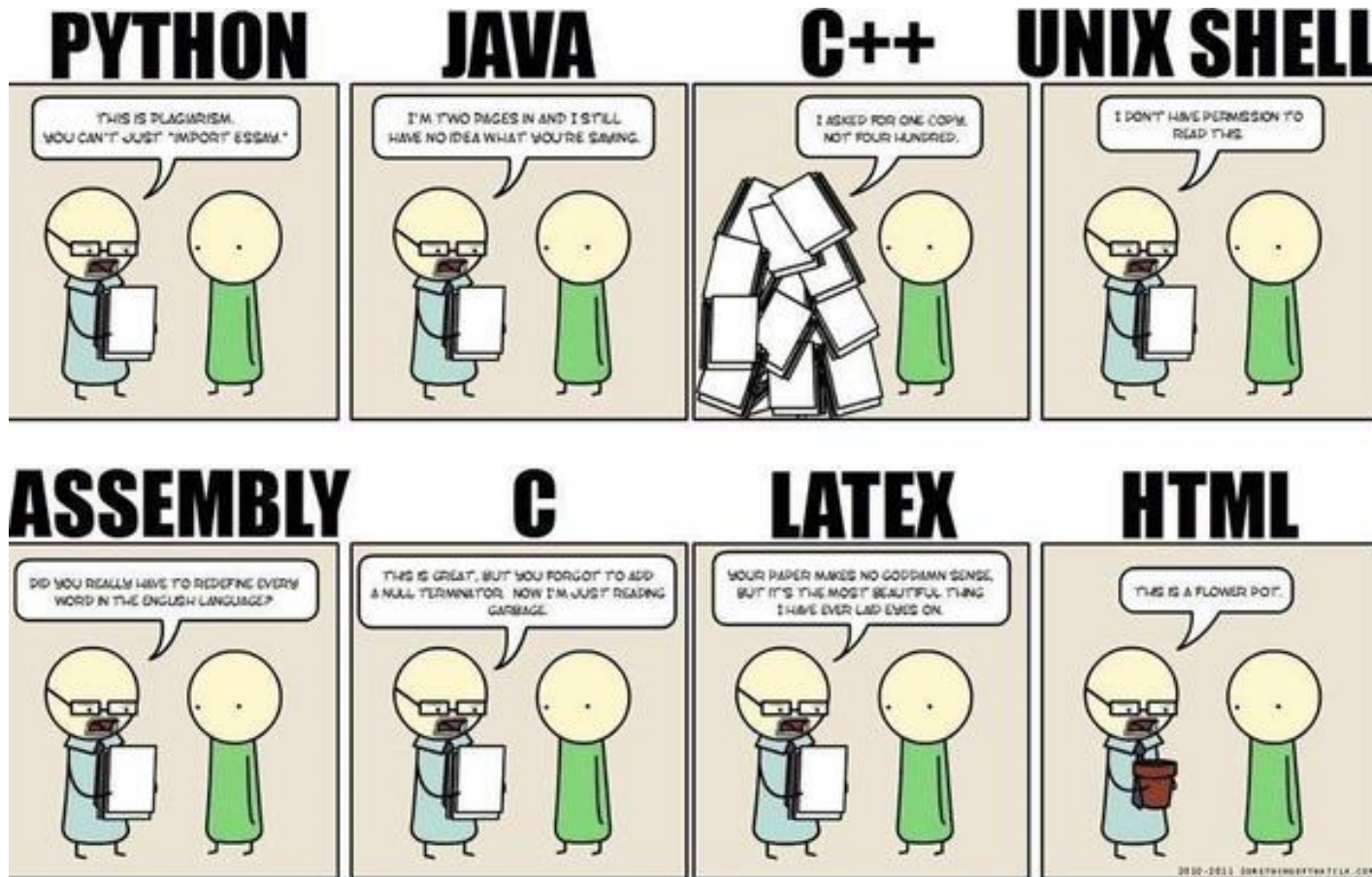
Step 6: Stop



10100110	01110110
00100110	00000000
11111010	11111010
01001110	10100110
11100110	10010110
11001110	00101110
10100110	01001110
11111010	01100110
01001110	10000110
etc...	



Programming language



Languages or algorithms



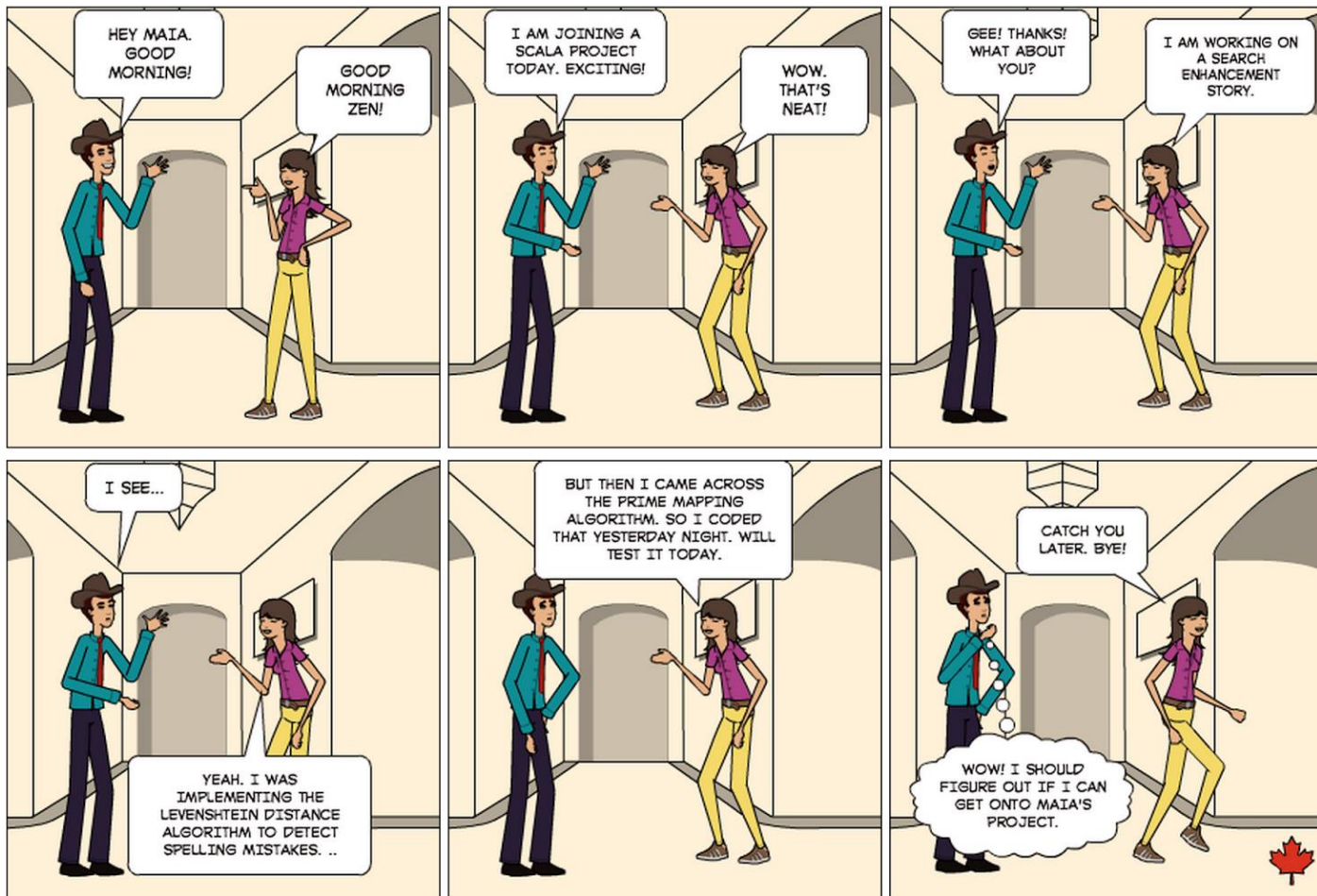
LANGUAGE OR ALGORITHM

by zenx

Monday January 19, 2015

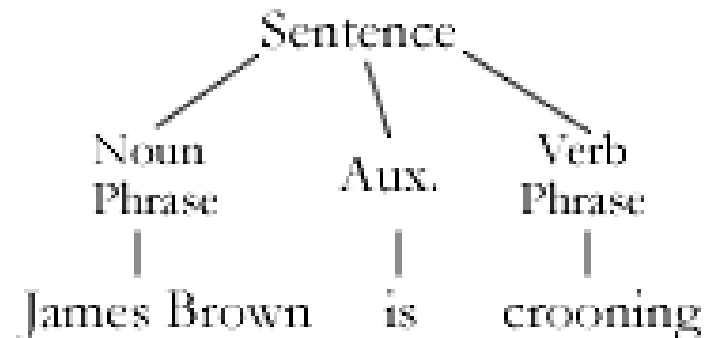
0 views | 0 comments

Lives developers live..



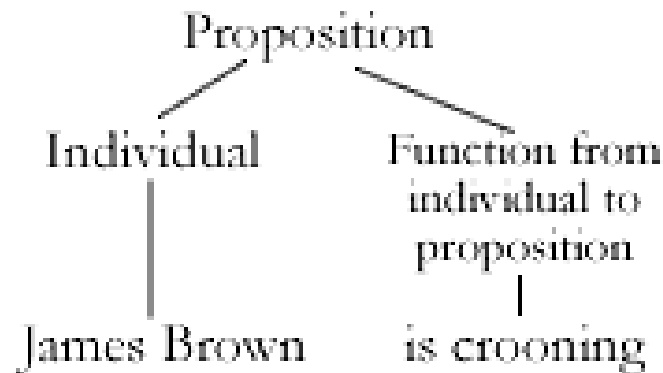
Syntax, Semantic and Translation

- The **syntax** of a language is a set of characters and the acceptable sequences of those characters.

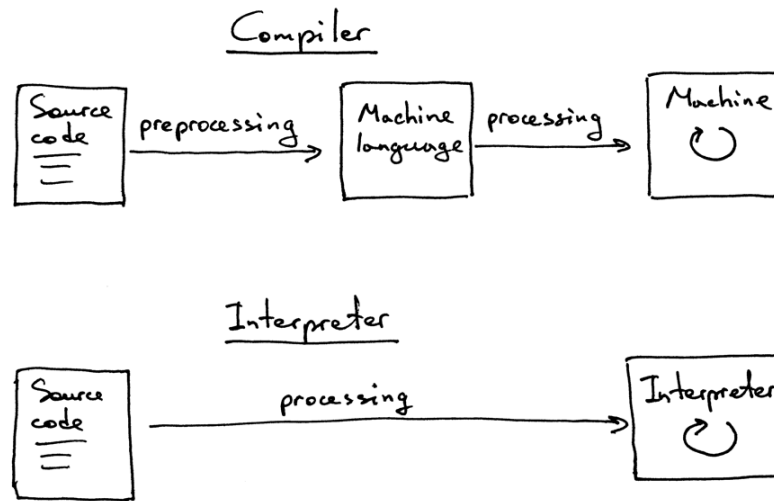


Syntax, Semantic and Translation

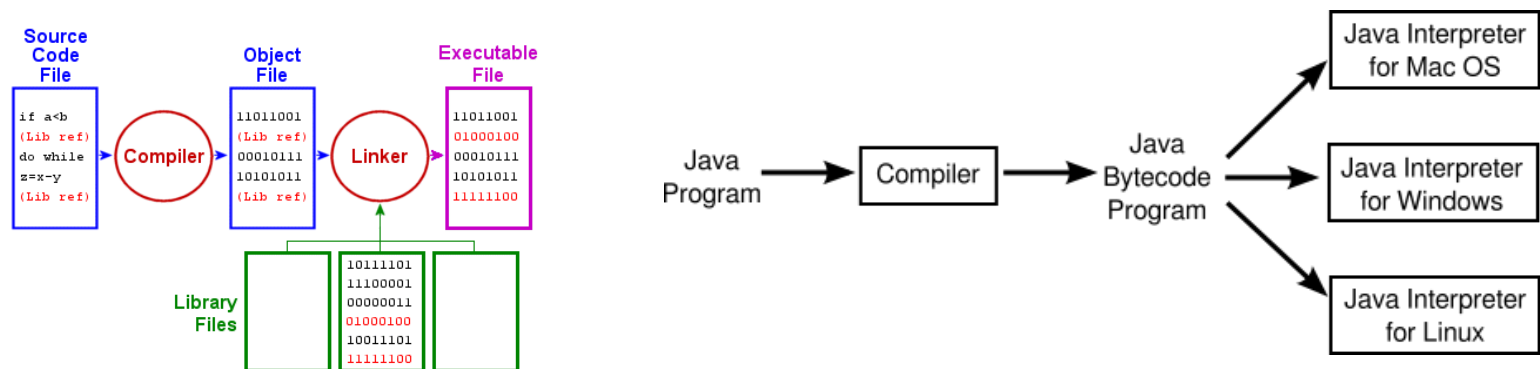
- The **semantics** of a language is the meaning associated with each syntactically correct sequence of characters.



Program translation



A **compiler** is a translator program that translates programs directly into machine code to be executed by the CPU. An **interpreter** executes program instructions in place of (“running on top of”) the CPU.



Errors

- Syntax error: violation of the programming language rules

The screenshot shows the Borland C++ IDE with the following code in `bcc1.cpp`:

```

1 bcc1.cpp
5 ink main()
6 {
7     cout << "Hello from BCC 5.5 and Scite!" << endl;
8 }

```

The command prompt shows the compilation command: `>bcc32 -v -w -O1 bcc1.cpp`. The output indicates the compiler version (Borland C++ 5.5.1) and then reports a syntax error:

```

Error: E011: bcc1.cpp: 5: Declaration syntax error
*** 1 errors in Compile ***

```

The error message is highlighted with a red box. The code line `ink main()` is the source of the error.

- Semantic errors: errors in the meaning



Round off errors and the Patriot missile

POSTED BY AUTAR KAW IN NUMERICAL ERRORS, UNCATEGORIZED

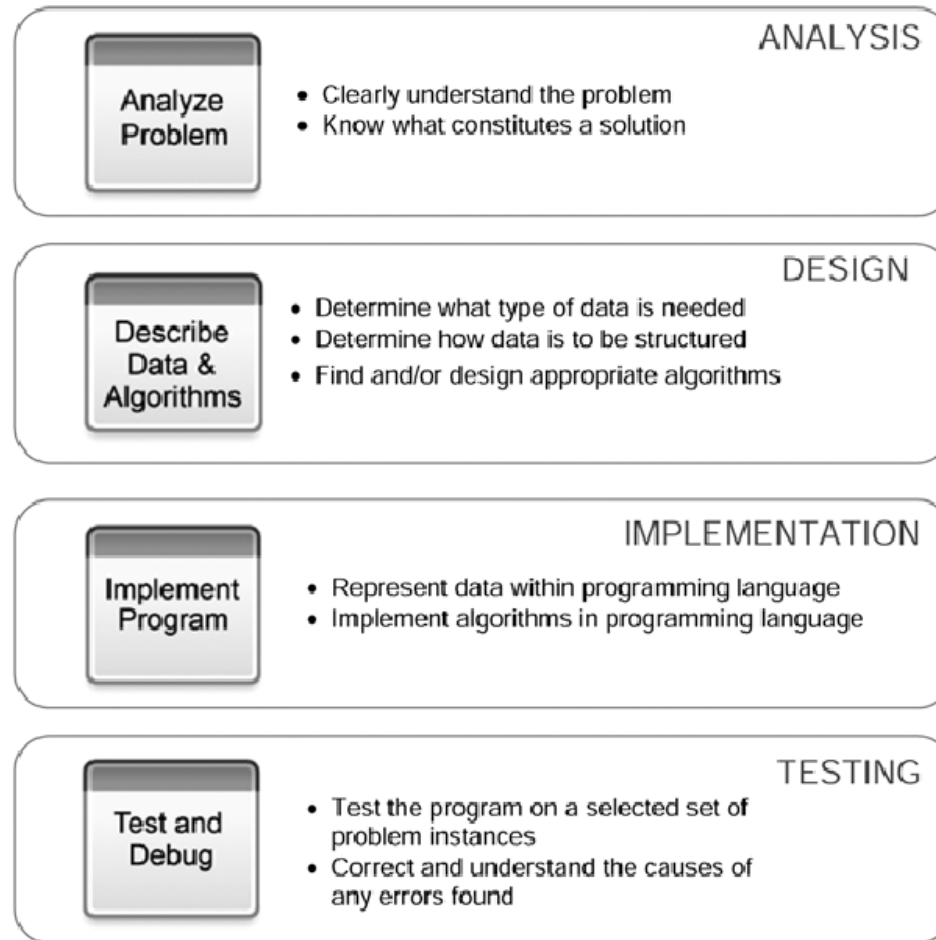
≈ 6 COMMENTS

★★★★★ 6 Votes

Twenty-eight Americans were killed on February 25, 1991 when an Iraqi Scud hit the Army barracks in Dhahran, Saudi Arabia. The Patriot defense system had failed to track and intercept the Scud. What was the cause for this failure?

The Patriot defense system consists of an electronic detection device called the range gate. It calculates the area in the air space where it should look for the target such as a Scud. To find out where the Patriot missile should be next, it calculates its location based on the velocity of the Scud and the last time the radar detected the Scud.

Computational Solving Problem



An example

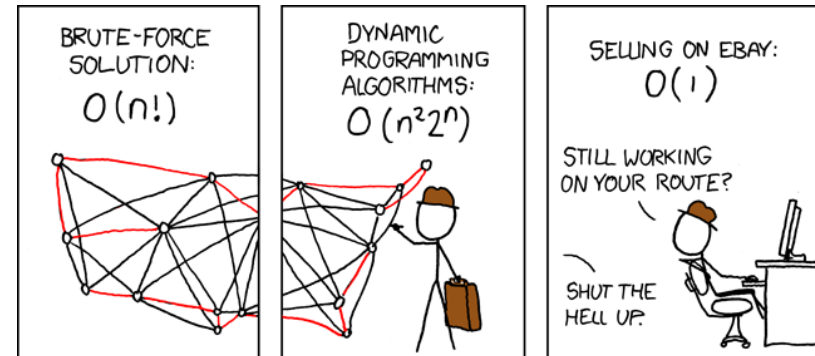
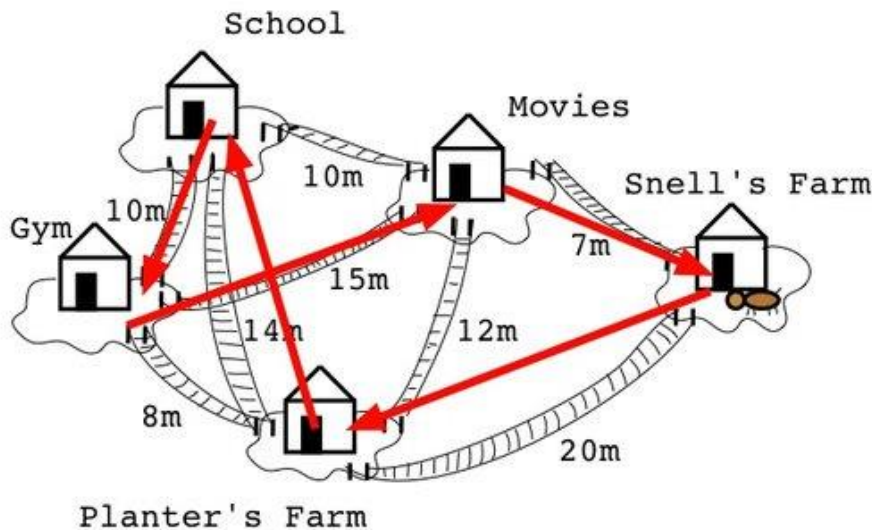
$$1+2+3+\dots+n=?$$



Let's start to program!!

- Create a .py file with your program
- Execute in python...

Limits of computational solving problem



Traveling salesman problem: Find the shortest route of travel for a salesman needing to visit a given set of cities

Any algorithm that correctly solves a given problem must **solve the problem in a reasonable amount of time**, otherwise it is of limited practical use.