Hochiminh City University of Technology
Computer Science and Engineering
[CO1027] - Fundamentals of C++ Programming

Introduction to Computer

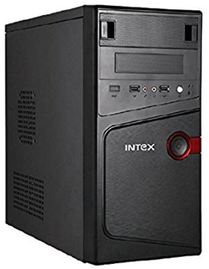
Lecturer: Duc Dung Nguyen
Credits: 3

Today's outline

- Hardware vs. Software
- Programming language
- Problem solving and software development
- Algorithm
- Discussion

Hardware vs. Software

- Hardware: physical components of computer (including peripherals)

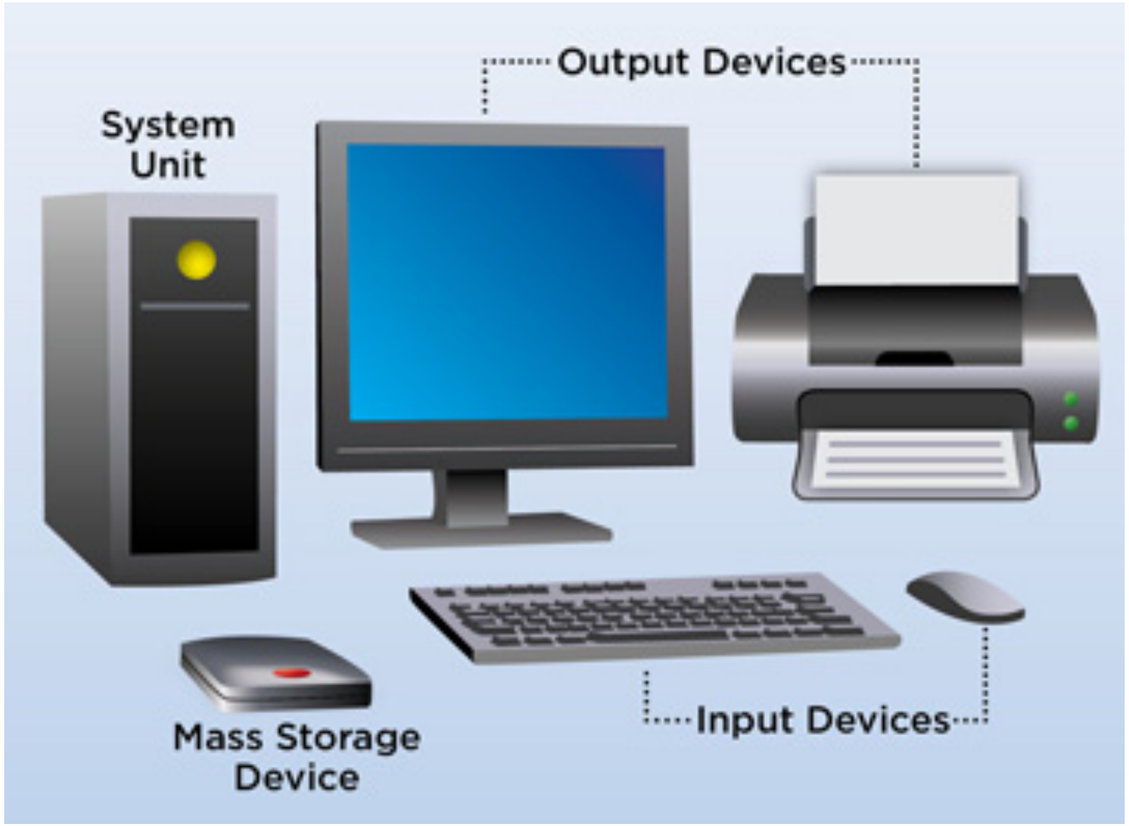




- Software: the programs that run on a computer



Computer Architecture

- Computer architecture/model
  - How computer runs
- Two types of architecture:
  - von Neumann
  - Harvard



von Neumann Architecture

- Created by scientist John von Neumann in 1945

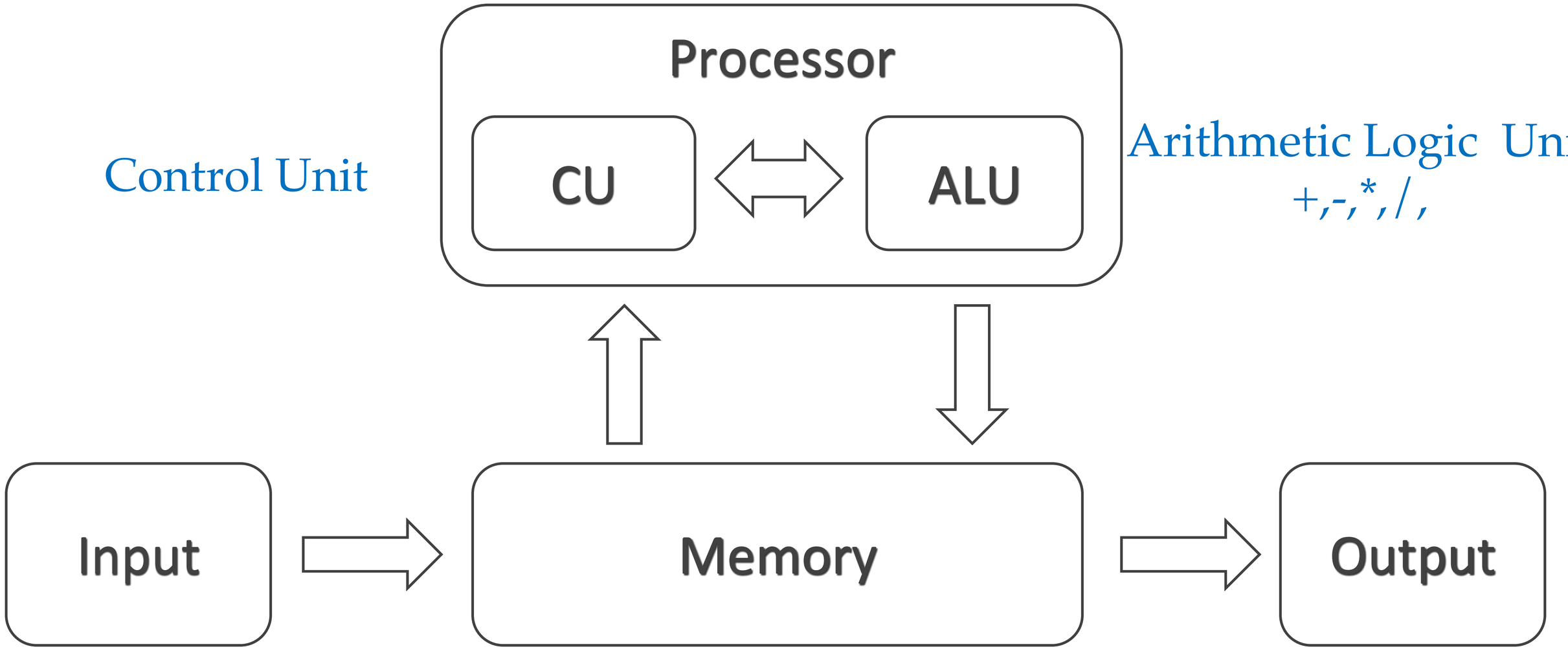
John von Neuman
(1903 – 1957)
American-Hungarian mathematician
Source: Wikipedia

- Uses on today's most computers

Mainframe

Personal computer

von Neumann Architecture



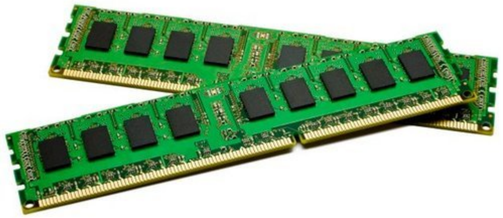
von Neumann Components

1) Input

keyboard

mouse

2) Memory

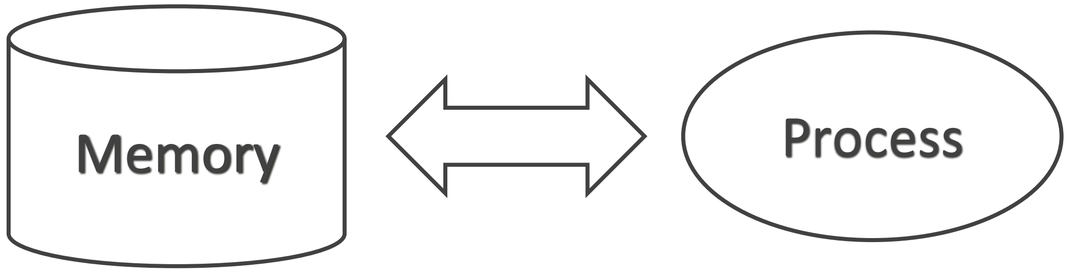


RAM

von Neumann Components (cont)

3) Processor

a) Control Unit
- Manage the process



b) Arithmetic Logic Unit
- Perform mathematics funcions
- Addition, subtraction, multiplication, division, etc

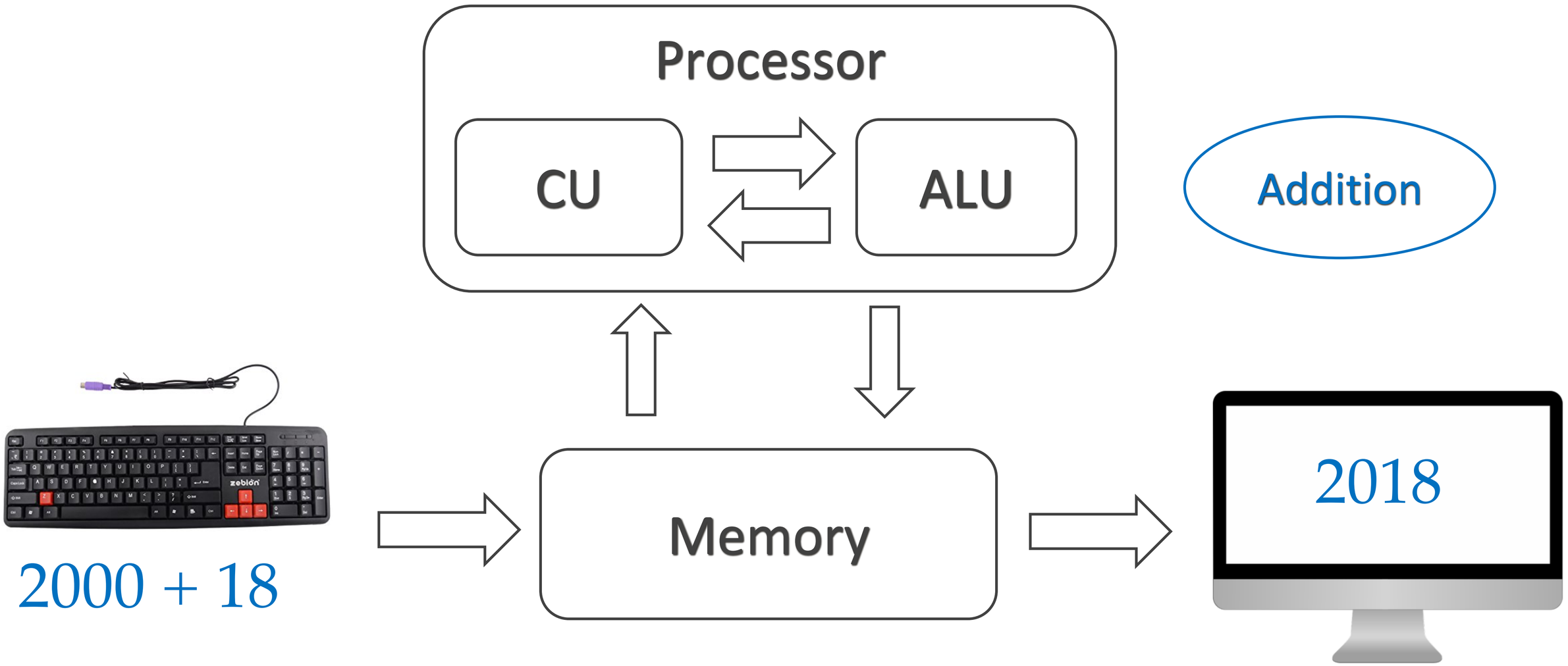
von Neumann Components (cont)

4) Output

monitor

printer

Example



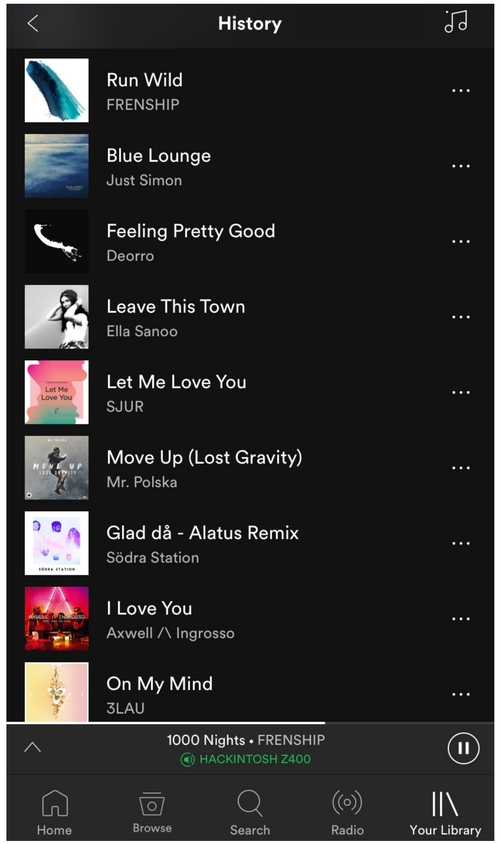
Software

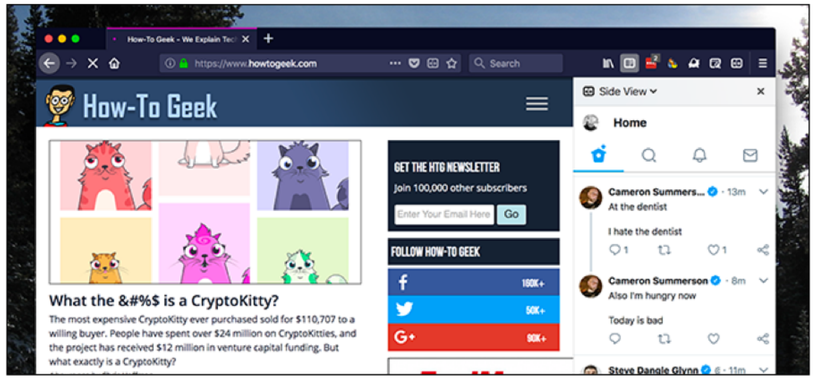
Software

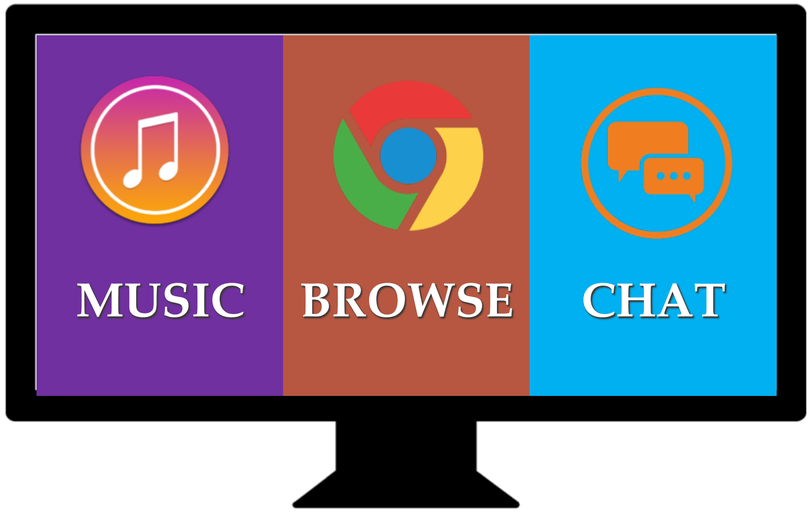
- Definition: A set of machine-readable instructions that directs a computer's processor to perform specific tasks. [Wikipedia]

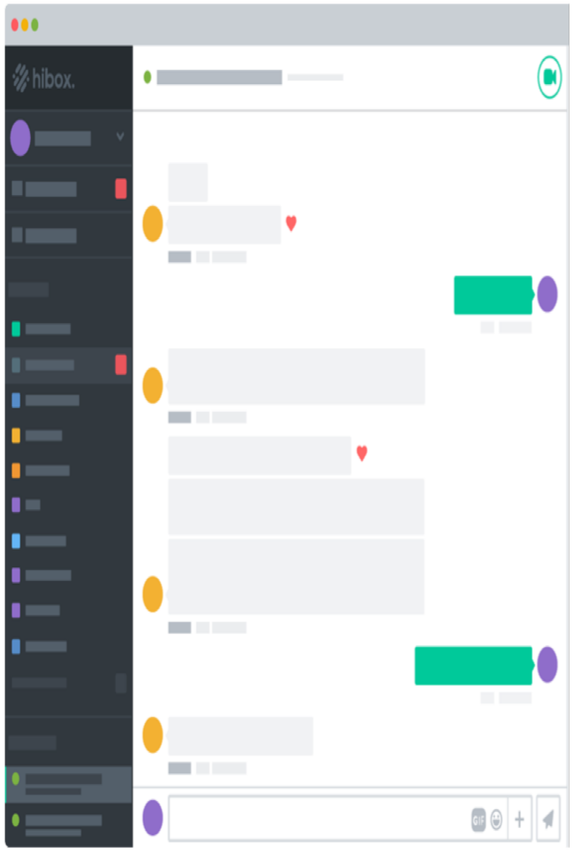
Software tells the CPU what to do

When multi-tasks, how?



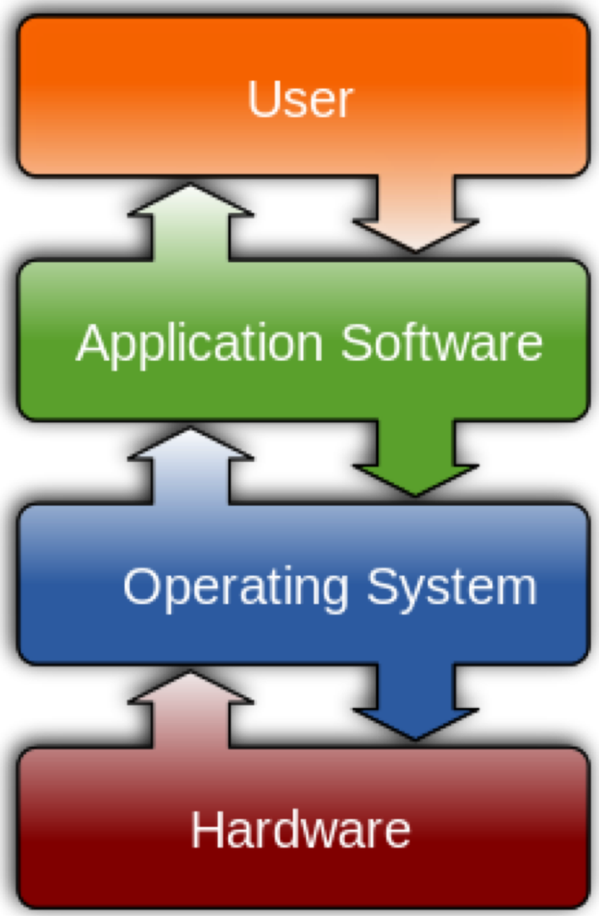






Operating system

Operating system: is the master program that manages how software gets to use the hardware of the computer

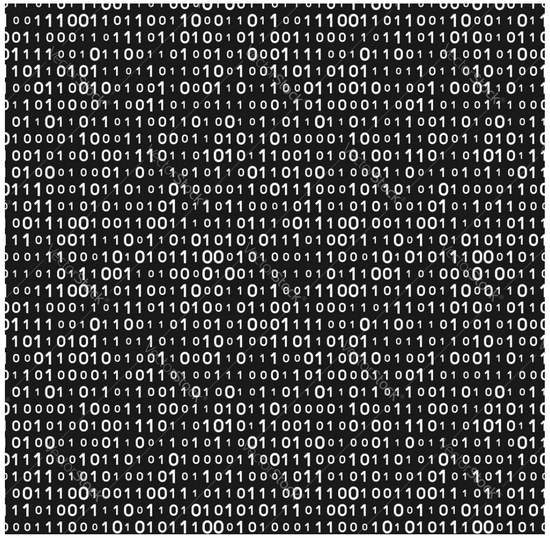


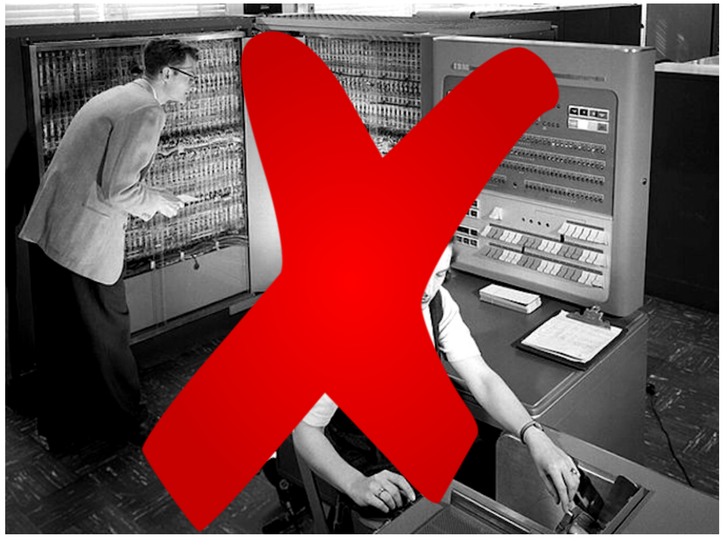
[source: Wikipedia]

Programming language

Binary code

- Binary code is the most basic form of software which can be executed on the computer.





High-level language

```cpp
int PointSet::inTri(int p1Idx, int p2Idx, int p3Idx, int pIdx) {
    // Check if a triangle is degenerate or not
    int s4 = testLeftTurn(p1Idx, p2Idx, p3Idx);
    if (s4 == 0)
        return 0; // Degenerate. Actually this is not a triangle

    int s1 = testLeftTurn(p1Idx, p2Idx, pIdx);
    int s2 = testLeftTurn(p2Idx, p3Idx, pIdx);
    int s3 = testLeftTurn(p3Idx, p1Idx, pIdx);


    if((s1 == 0 && s2 == 0) || (s1 == 0 && s3 == 0) || (s2 == 0 && s3 == 0))
        return 1; // This point is on one of the vertices of the triangle
    else if(s1 == 0) {// This point is on the straight line 12
        if(s2 == s3)
            return 1; // This point is on the edge 12
        else
            return -1;
    }
    else if(s2 == 0) {// This point is on the straight line 23
        if(s1 == s3)
            return 1; // This point is on the edge 23
        else
            return -1;
    }
    else if(s3 == 0) {// This point is on the straightt line 31
        if(s1 == s2)
```

C++

```python
class LintRunner(object):
    """ Base class provides common functionality to run
          python code checkers. """

    sane_default_ignore_codes = set([])
    command = None
    output_matcher = None

    custom_ignore_codes = None
    # flymake: ("\\(.*\\) at \\([^ \n]+\\) line \\([0-9]+\\)[,.\n]" 2 3 nil 1)
    # or in non-retardate: r'(.*) at ([^ \n]) line ([0-9])[,.\n]'
    output_format = ("%(level)s %(tool)s/%(error_type)s%(error_number)s:"
                     "%(description)s at %(filename)s line %(line_number)s.")

    def __init__(self, config):
        self.config = config
        if self.config.VIRTUALENV:
            # This is the least we can get away with (hopefully).
            self.env = {
                'VIRTUAL_ENV': self.config.VIRTUALENV,
                'PATH': self.config.VIRTUALENV + '/bin:' + os.environ['PATH']}
        else:
            self.env = {}

        self.env.update(self.config.ENV)

    @property
    def operative_ignore_codes(self):
        ignore_codes = self.config.IGNORE_CODES
        ignore_codes |= set(getattr(self.config, self.custom_ignore_codes))
        if self.config.USE_SANE_DEFAULTS:
            return ignore_codes ^ self.sane_default_ignore_codes
```

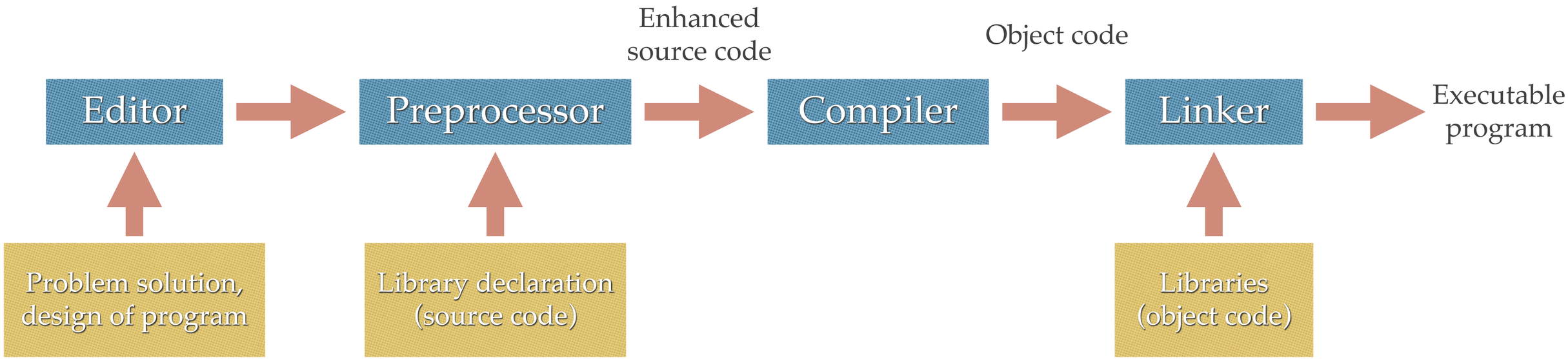
Python

Programming language

- The high-level language code is called source code.
- The compiled machine language code is called target code.
- Compiler translates the source code into target machine language.
- Tools for development
  - Editor: support text editing feature for writing source code
  - Preprocessor, compiler, linker: tools for translating source code
  - Debugger: traces program's execution in order to locate bugs!
  - Profiler: collects statistics about a program's execution

Programming language

- The process of building a computer program



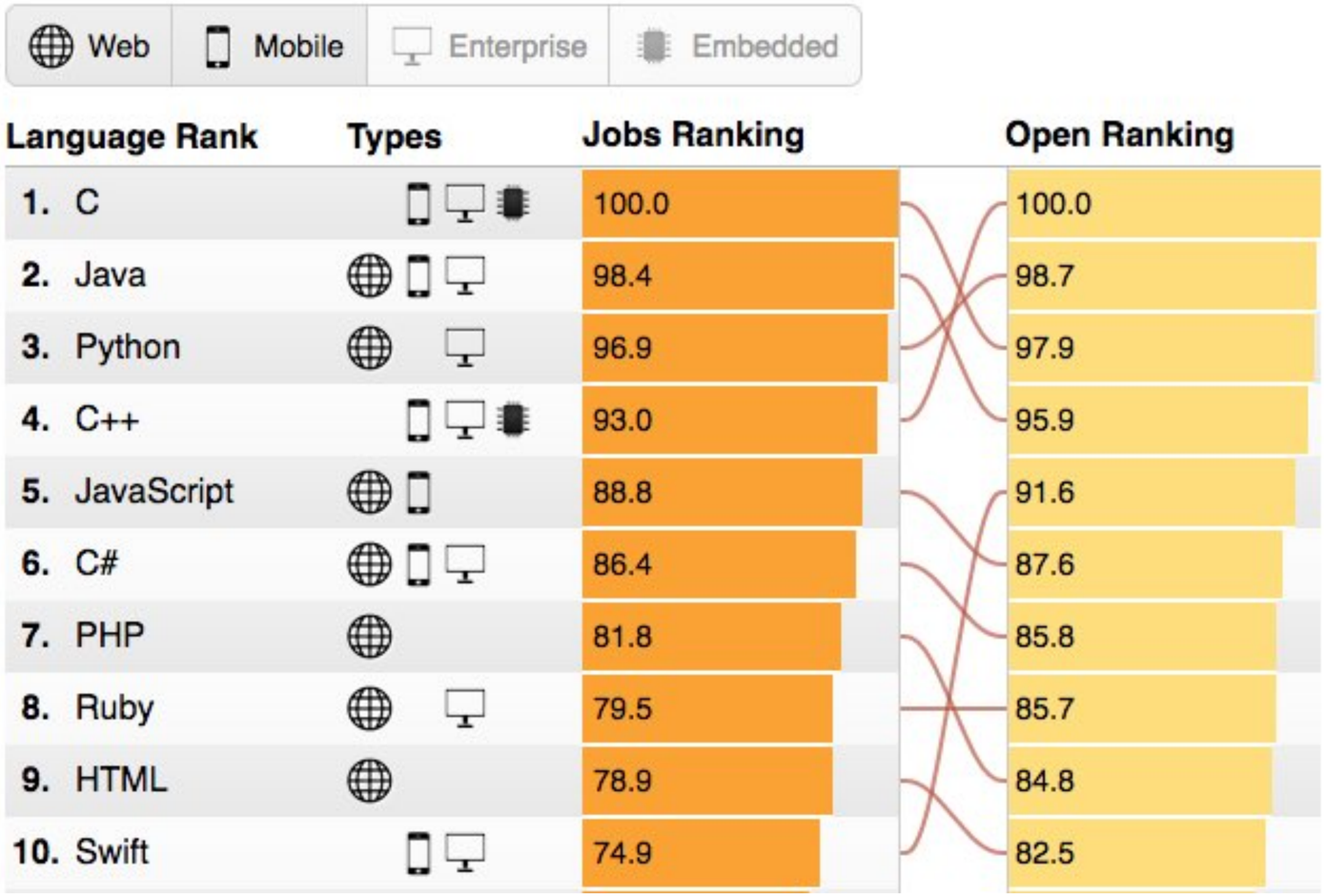
Programming language

https://fossbytes.com/programming-language-most-important-2016-ieee/

Which language to use?

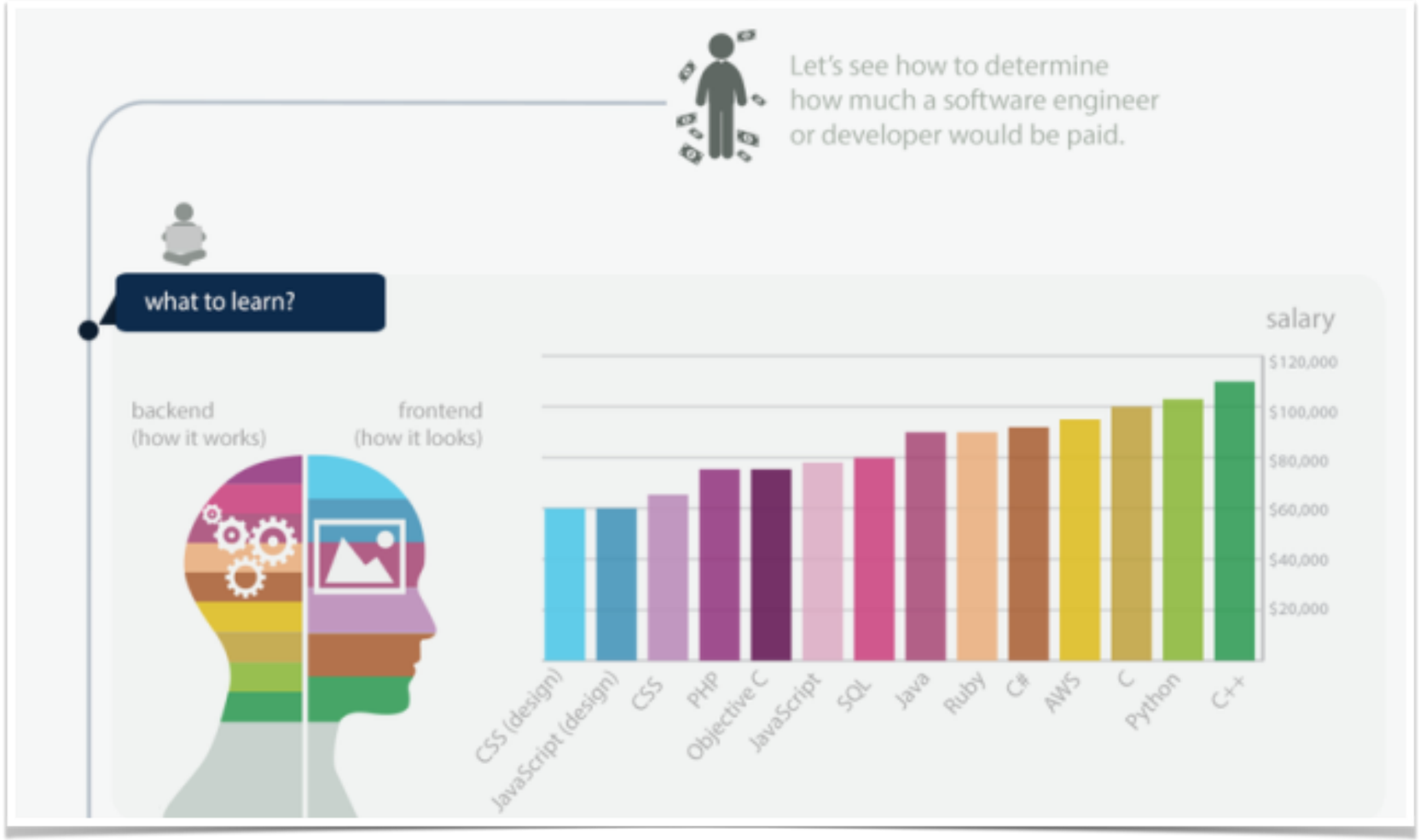
- The language must be fit-for-purpose
- The choice of platform is critical
- Existing skills of the programmers

Which language to use?



https://fossbytes.com/programming-language-most-important-2016-ieee/

Why is C/C++?



[source: lifehacker]

Why is C/C++?

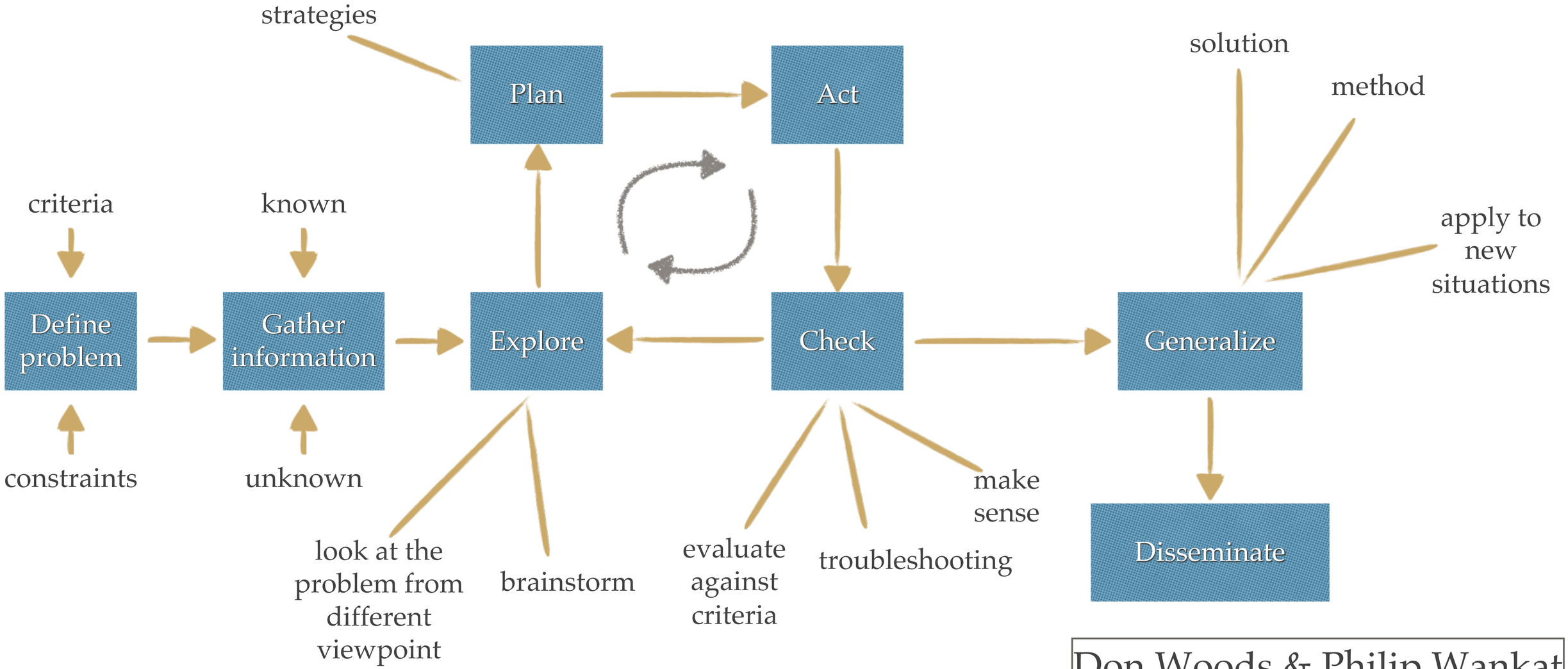
- The language that places a foundation for many others
- Gain high-level programming skills
- Freedom, flexibility, advanced techniques, …

Problem solving

Problem solving

- Think about how you solved your problems before
  - Math, Physic, Chemistry, etc.
- How did you solved them?
  - Systematic
  - Random idea
  - Memorising solutions
  - Other approaches

Problem solving



Don Woods & Philip Wankat

Problem solving

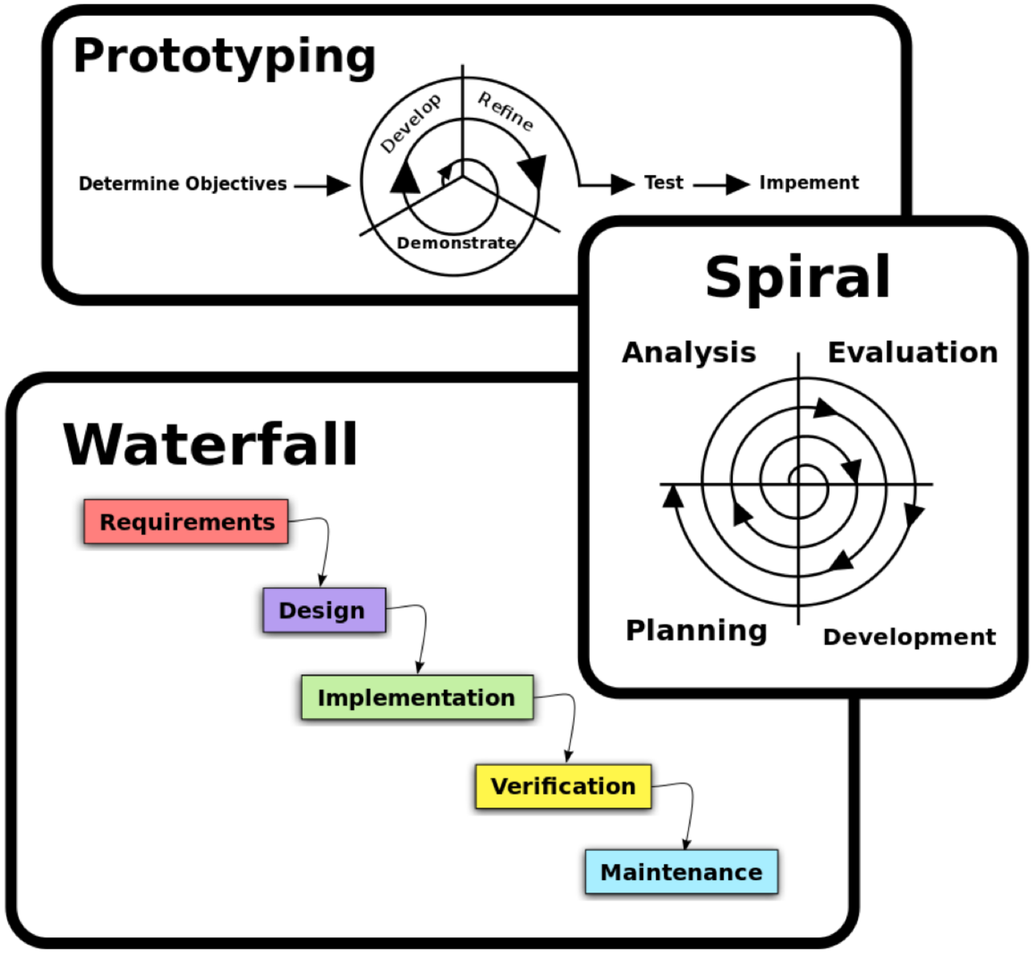
- How will you solve a problem on computer?
  - Computer deals best with performing easy tasks over and over again.
- Principle: break the big problem into smaller pieces
  - We utilize the computer's ability by implementing repetitive techniques to incrementally solve our complex problems.
- Think about a problem that you can solve using this method.

Problem solving

- Two basic approaches
  - Iterative: solve a part of the problem, repeat the process on the remaining problem until the original problem is solved
  - Recursive: define how to solve simplest problems. Then we break the problem into simpler and simpler pieces until they reach the level that computer know how to solve (by our definition).

Software development

- The software development process follows the principles of problem solving.
- Various software development models existed
- What do you need to develop a software at the basic level?
  - Compiler, Debugger, Editor
  - Integrated Development Environment (IDE)
    - Visual Studio, Eclipse, Xcode, etc.



[source: Wikipedia]

Software development

- Address your problem, collect requirements
- Analyse feasible approaches, find solution
- Design algorithm
- Implement: write code
- Compile, debug
- Evaluate the program
- Deploy software

Steps in development of algorithm

- Problem definition
- Development of a model
- Specification of Algorithm
- Designing an Algorithm
- Checking the correctness of Algorithm
- Analysis of Algorithm
- Implementation of Algorithm
- Program testing
- Documentation Preparation

Algorithm

- Algorithm is a self-contained list of operations to be performed.
- An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function.
- Describe algorithm
  - Text: details of data flow and processing steps
  - Pseudo code
  - Flowchart

Algorithm

- Pseudo code
  - Independent from programming language.
  - An informal high-level description of the operating principle of a computer program or algorithm.
  - No standard for pseudo code syntax
  - Algorithm is often designed with pseudo code description

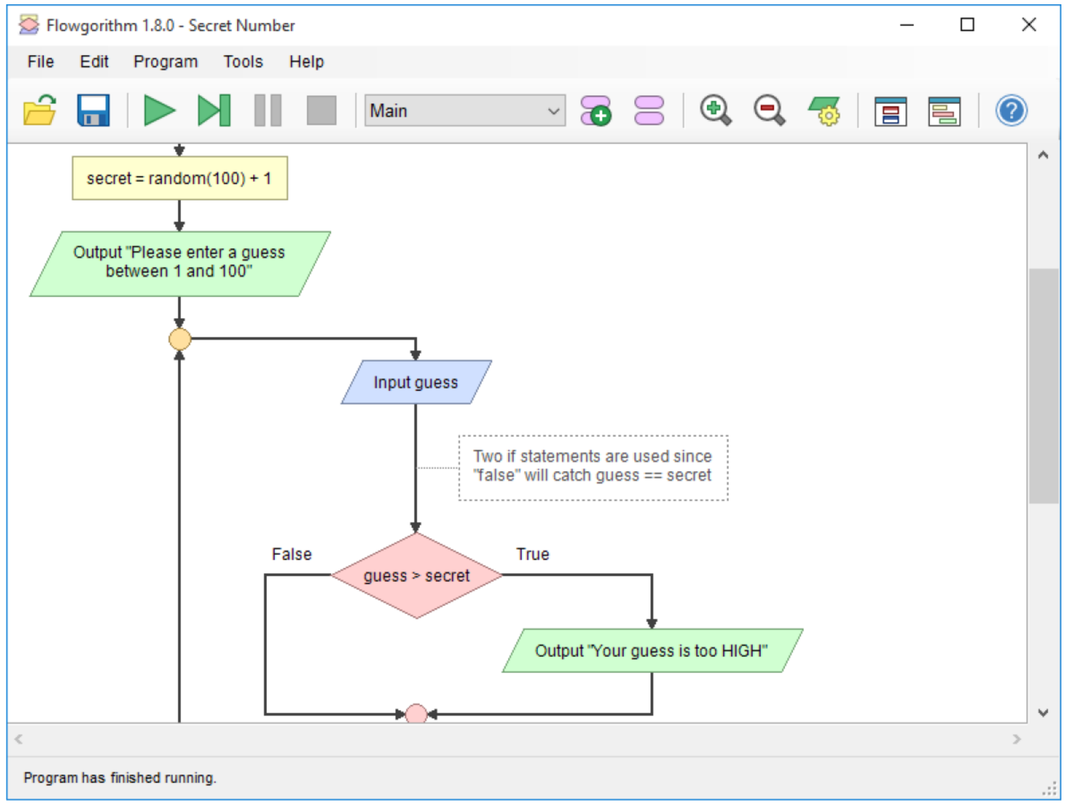
```
function randomSound()
Loop: from i = 1 to 100
    print_number = True
    If i is even Then
       If print_number Then
         Print (i + random())
       Else
         PlaySound( rand() )
       End If
    Else
       Print “Odd…”
    End If
End (loop)
End (function)
```

Algorithm

- Pseudo code - guideline
  - Mimic good code and natural language
  - Ignore unnecessary details
  - Don’t belabour the obvious
  - Take advantage of programming shorthands
  - Consider context
  - Don’t lose sign of the underlying model
  - Check for balance

Algorithm

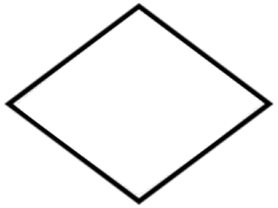
- Flow chart
  - A type of diagram that represents the algorithm (in our context). The flow chart illustrates a solution model to a given problem.
  - A flow chart is constructed from basic shapes that have specific meanings.



Algorithm

- Flow chart - Building blocks


Flow line

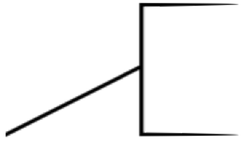

Decision


Process


On-page connector


Input/Output


Preparation


Annotation


Predefined process


Off-page connector


Terminal

[source: Wikipedia]

Algorithm

- Flowchart
  - Flow line: represents control pass from one symbol to another.
  - On-page connector: has more than one arrow coming into it but only one going out. It is useful to represent iterative processes.
  - Annotation: represents comments or remarks about the flowchart.
  - Terminal: usually has word/phrase to indicate the start/end of a process.
  - Decision: where the decision must be made (usually Y/N).

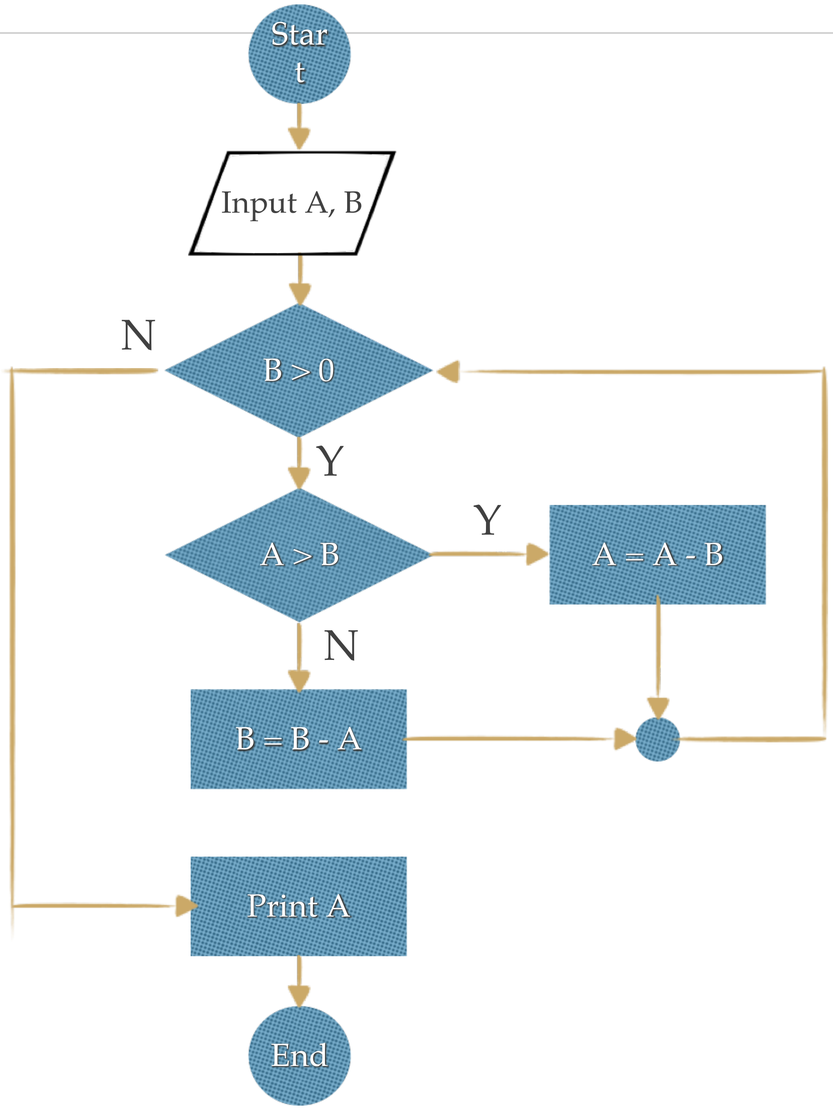
Algorithm

- Flowchart
  - Input/Output: involves receiving data and displaying processed data.
  - Predefined process: represents complex processing steps which maybe detailed in a separate flowchart.
  - Process: shows that something is performed.
  - Preparation: prepares a value for a subsequent conditional or decision step (replace decision symbol in case of conditional loop).
  - Off-page connector: connect to another page.

Algorithm

- Flowchart - example

```
INPUT A,B
Loop while B > 0
    If A > B then
        A = A - B
    Else
        B = B - A
    End
End loop
Print A
```



Summarise

- Understand concepts of Hardware/Software
- Understand problem solving process
- Definition of algorithm
- Role of algorithm in problem solving process
- Knowing how to describe an algorithm
- Understand concepts of pseudocode and flowchart