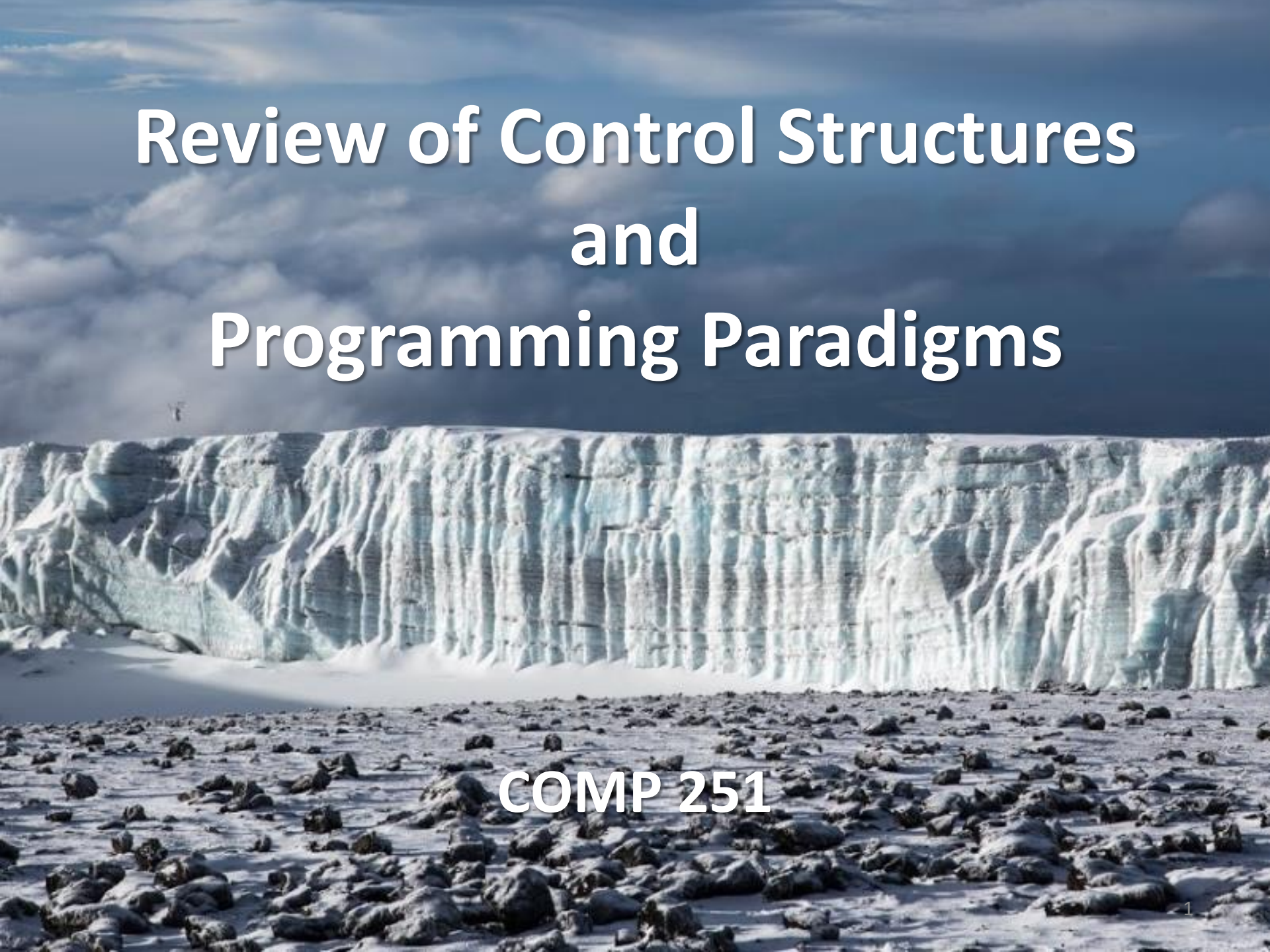


Review of Control Structures and Programming Paradigms

COMP 251

Goals of this Lecture

Since this course is about Data Structures *and* Algorithms, then let us review the following fundamental concepts involving the latter; namely,

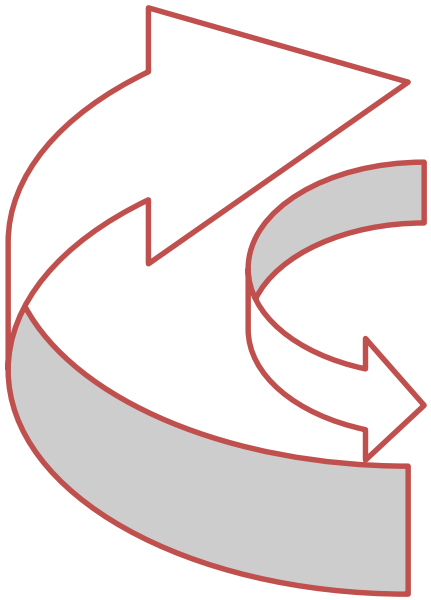
- The Flow of Logic,
- The Classification of Algorithms, and
- Algorithmic Paradigms

The Flow of Logic

Dijkstra’s Control Structures

Pre-60's Programming: The GOTO Statement

Consider this sequence of instructions

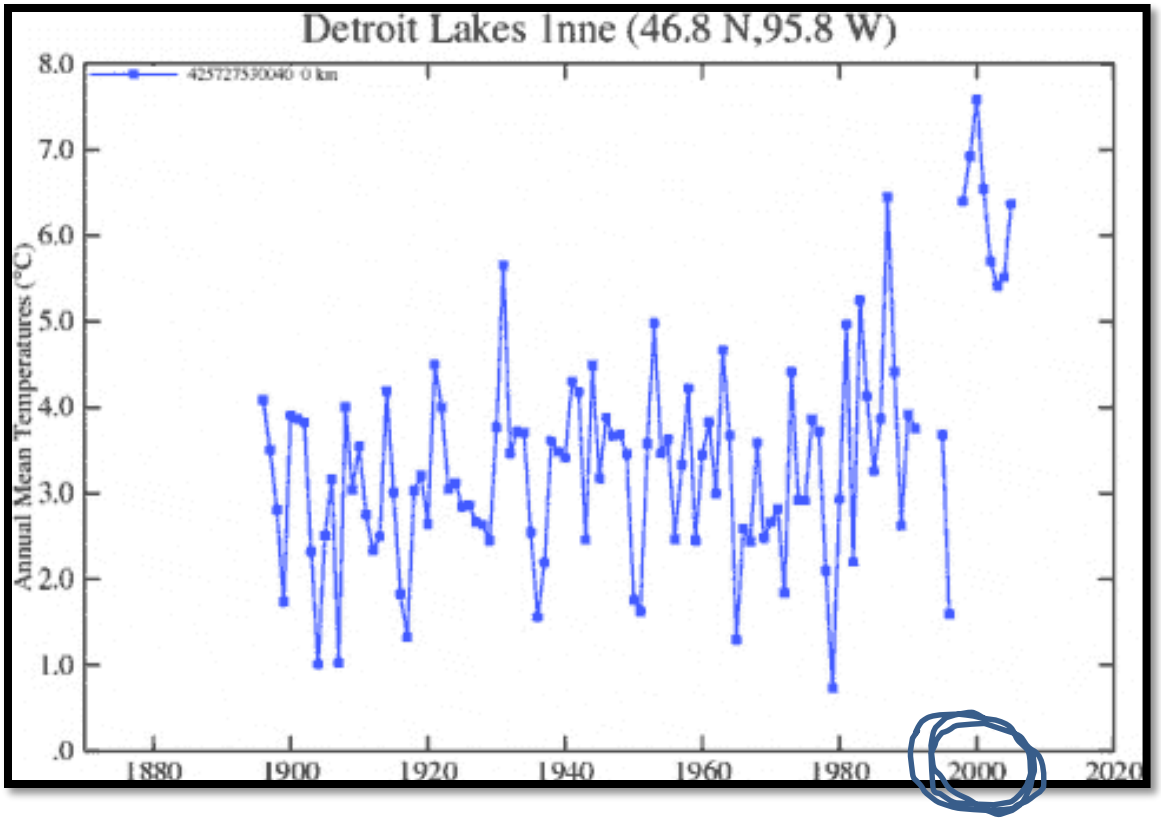


- …
- Statement 1
- If … goto 4
- Statement 3
- Statement 4
- If … goto 1
- Statement 6
- …

Informal Definition: Spaghetti Code

If a code is written with little attention paid to the flow of logic, it tends to be unstructured and largely TANGLED. Such code is colloquially refer to this as “spaghetti code.”

Y2K



http://www.dailytech.com/Blogger+finds+Y2K+bug+in+NASA+Climate+Data/article8383.htm

A *recipe* is a **sequence** of instructions

- One desires an efficient flow of logic (flow of execution).

A *recipe* is a **sequence** of instructions

- One desires an efficient flow of logic (flow of execution).
- The flow of logic can be controlled by certain characteristics of algorithms.

A *recipe* is a **sequence** of instructions

- One desires an efficient flow of logic (flow of execution).
- The flow of logic can be controlled by certain characteristics of algorithms.
- **In particular, algorithms often have steps that <u>repeat</u> or require <u>decisions</u> as well as simple sequences of instructions.**

A *recipe* is a **sequence** of instructions

- Making pancakes:
  - Gather ingredients.
  - Beat egg until homogeneous.
  - Mix in water, oil and cereal until homogeneous.
  - Pour resulting batter onto hot, oiled pan using ¼ cup batter for each cake.
  - Cook until bubbles form on surface, edges look dry and underside is golden brown.
  - Turn and brown otherside.
  - Serve with desired topping.

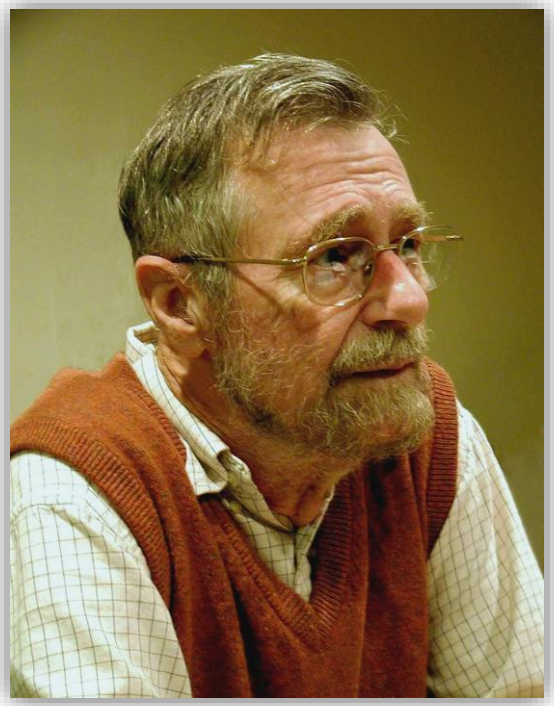
A *recipe* is a **sequence** of instructions

- Making pancakes:
  - Gather ingredients.
  - **Beat** egg **until** homogeneous.
  - **Mix** in water, oil and cereal **until** homogeneous.
  - Pour resulting batter onto hot, oiled pan using ¼ cup batter for each cake.
  - **Cook until** bubbles form on surface, edges look dry and underside is golden brown.
  - Turn and brown otherside.
  - Serve with desired topping.

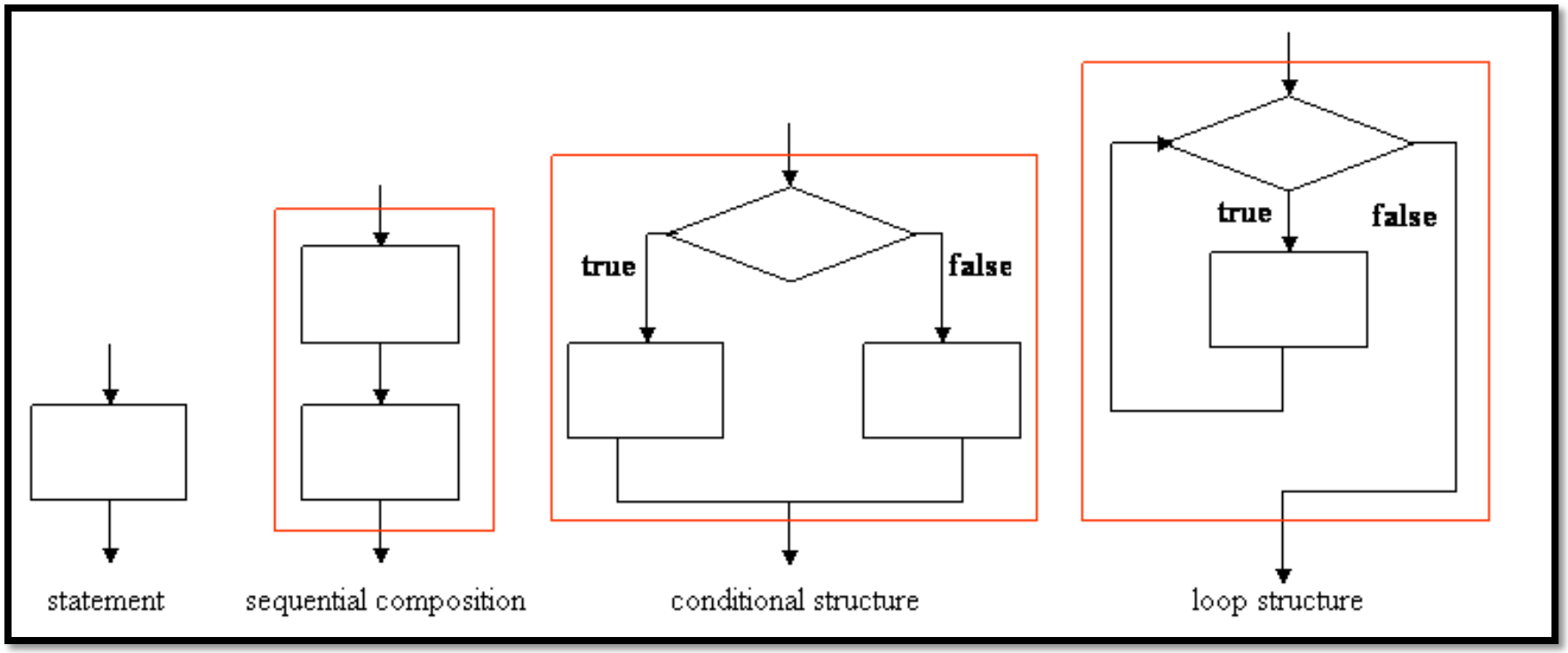
Edsger W. Dijkstra
1930–2002

Dijkstra (and others) in the late 1960's recognized that the *flow of logic* in code can be organized according to three primary **CONTROL STRUCTURES**

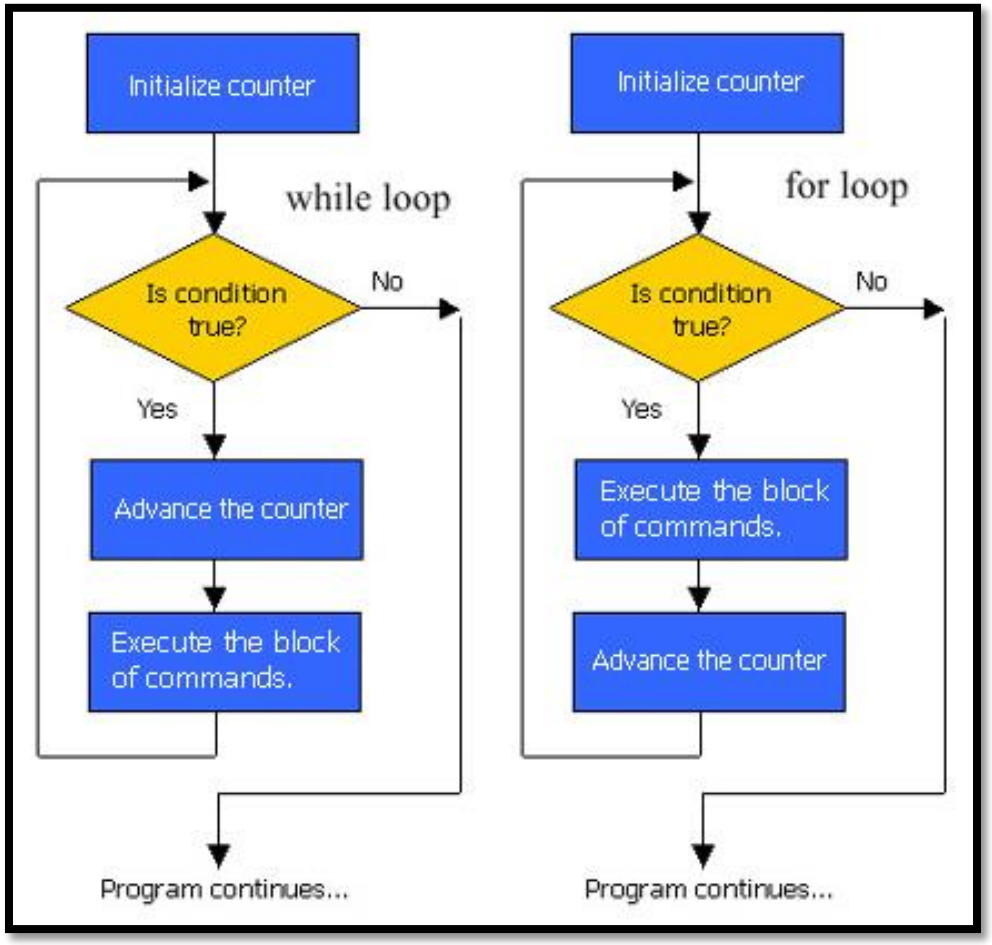
1. Sequence
2. Iteration (repeating a procedure)
3. Decision



Flow Chart Description



NOTE: Different Loop Structures



While-Loop Versus For-Loop

- **For-loops** are used when you know exactly how many iterations are involved.

While-Loop Versus For-Loop

- **For-loops** are used when you know exactly how many iterations are involved.
- **While-loops** are more flexible (iterate until a condition is met, for example).

While-Loop Versus For-Loop

- **For-loops** are used when you know exactly how many iterations are involved.
- **While-loops** are more flexible (iterate until a condition is met, for example).
- These are also classified as *iterative loops* (for-loop) and *conditional loops* (while-loop).

Classification of Algorithms

Related to Hardware

Classification by “Abstraction” from Hardware

- Computers are binary and digital at their core.

Classification by “Abstraction” from Hardware

- Computers are binary and digital at their core.
- Hence, “machine language” is pure binary.

Classification by “Abstraction” from Hardware

- Computers are binary and digital at their core.
- Hence, “machine language” is pure binary.
- The first step away from machine language towards natural language is Assembly.

Classification by “Abstraction” from Hardware

- Computers are binary and digital at their core.
- Hence, “machine language” is pure binary.
- The first step away from machine language towards natural language is Assembly.
- This is utilized by computer designers and engineers.

Classification by “Abstraction” from Hardware

- Computers are binary and digital at their core.
- Hence, “machine language” is pure binary.
- The first step away from machine language towards natural language is Assembly.
- This is utilized by computer designers and engineers.
- The next step away is a so-called “high-level language”, such as Java.

Calculating the nth Fibonacci Number

Assembly Language

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1

    @@:
        lea eax, [ebx
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

```
unsigned int fib(unsigned int n)
{
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        int a,b,c;
        a = 1;
        b = 1;
        while (true) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

High-Level Programming Language

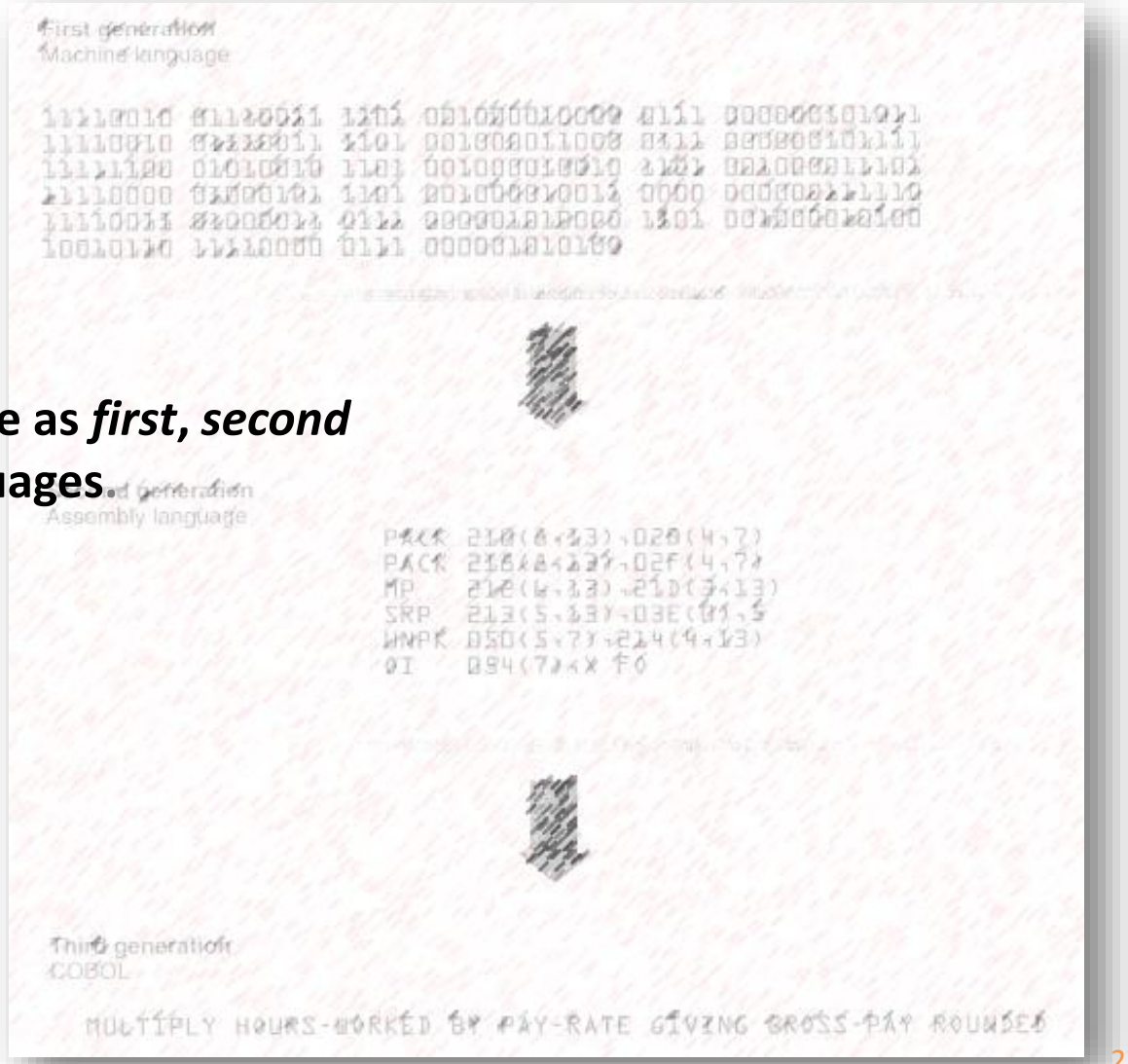
Machine Language

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD98B
C84AEBF1 5BC3
```

Classification by “Abstraction” from Hardware

Classification by “Abstraction” from Hardware

Note that we refer to these as *first*, *second* and *third* generation languages.



Classification by “Abstraction” from Hardware

- **First Generation** is *machine-level*.

Classification by “Abstraction” from Hardware

- **First Generation** is *machine-level*.
- **Second Generation** – Assembly, is still *not* considered a programming language since it possess *no semantics* but, instead, possesses a nearly injective (one-to-one) relationship with machine architecture.

Classification by “Abstraction” from Hardware

- **First Generation** is *machine-level*.
- **Second Generation** – Assembly, is still *not* considered a programming language since it possess *no semantics* but, instead, possesses a nearly injective relationship with machine architecture.
- **Third Generation** – nearly hardware independent.

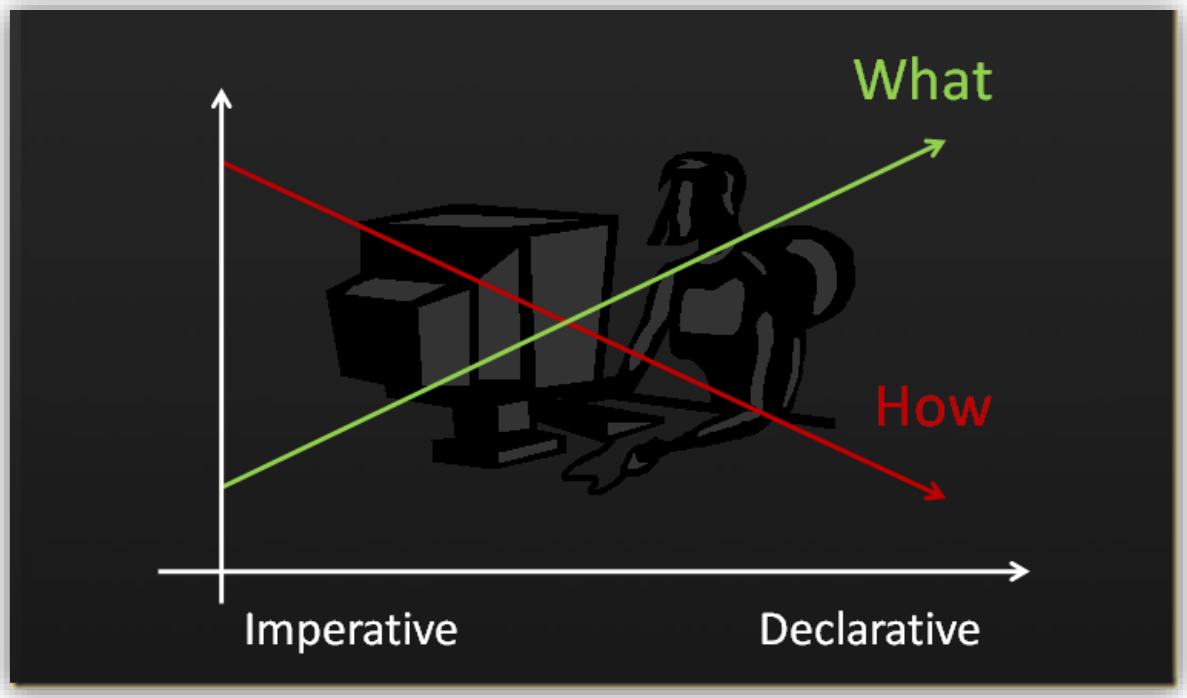
Classification by “Abstraction” from Hardware

- **First Generation** is *machine-level*.
- **Second Generation** – Assembly, is still not considered a programming language since it possess *no semantics* but, instead, possesses a nearly injective relationship with machine architecture.
- **Third Generation** – nearly hardware independent.
- **Fourth and Fifth Generation** – confused terminology and later abandoned.

Classification of Algorithms by Implementation

Programming Paradigms
(Fundamental Abstraction)

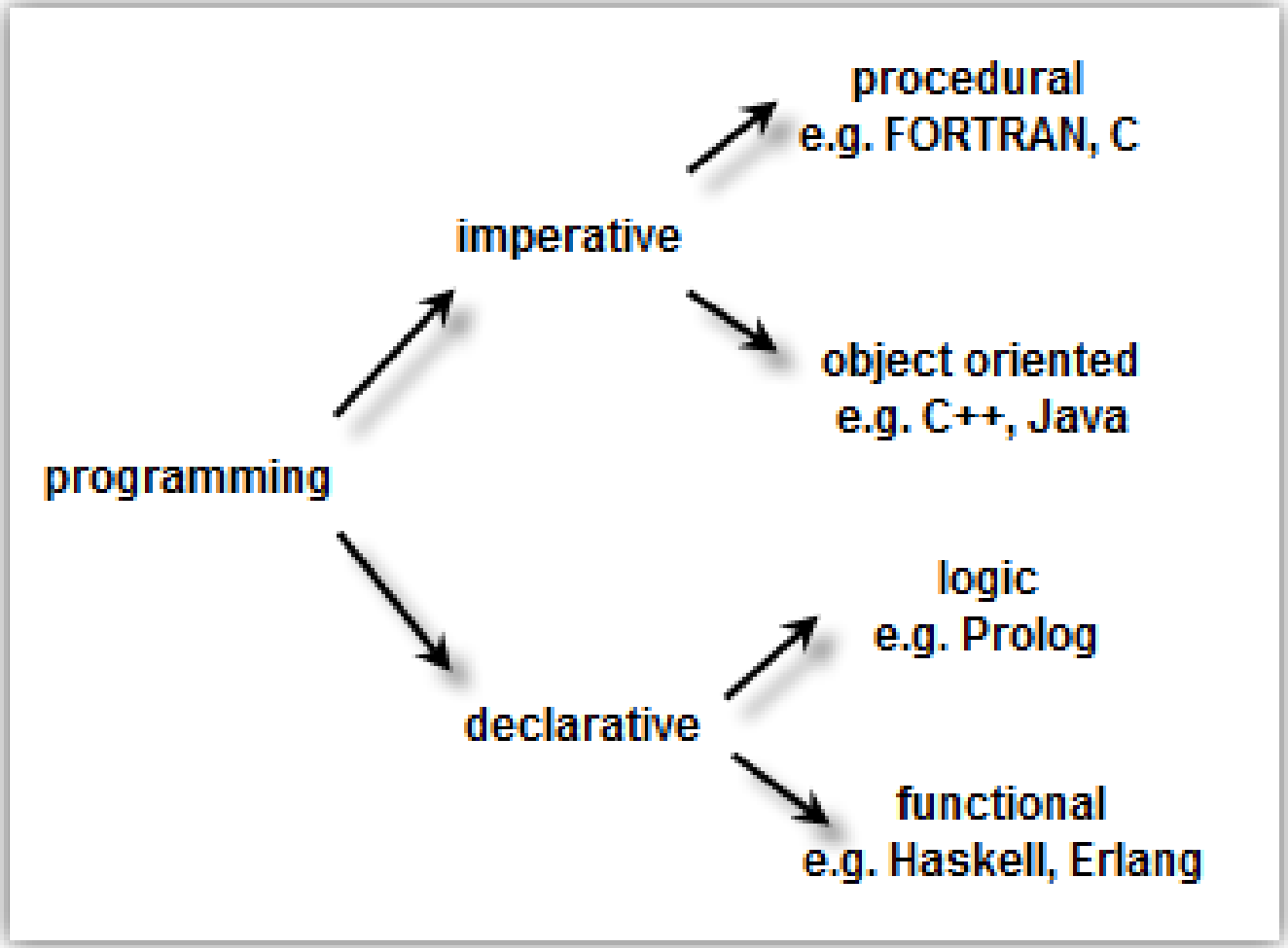
Implementation Classification



There are two fundamental **programming paradigms**:

- **Declarative programming** centers on **what** computation should be performed.
- **Imperative programming centers on how** to compute the problem *explicitly*.

Implementation Classification



Imperative

- **Imperative** specifies the sequence of operations to perform in order to achieve a desired result.
  - **procedural** uses procedures, such as functions, to describe the commands the computer should perform.
  - **OO** represents the concept of objects that have data fields (descriptive attributes of objects) and associated procedures known as methods.

Wikipedia

Declarative

- **Declarative** specifies the desired result, rather than how to achieve it.
  - **Logic programming** is based upon formal logic (usually first order predicate logic).
  - **Functional programming** emphasizes functions that produce results that depend only on their inputs on not on the program state.

Wikipedia

Examples of imperative and declarative code

Taking a simple example, let's say we wish to double all the numbers in an array.

We could do this in an imperative style like so:

```
var numbers = [1,2,3,4,5]
var doubled = []

for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled) //=> [2,4,6,8,10]
```

We explicitly iterate over the length of the array, pull each element out of the array, double it, and add the doubled value to the new array, mutating the `doubled` array at each step until we are done.

From http://latentflip.com/imperative-vs-declarative/

A more declarative approach might use the `Array.map` function and look like:

```
var numbers = [1,2,3,4,5]

var doubled = numbers.map(function(n) {
  return n * 2
})
console.log(doubled) //=> [2,4,6,8,10]
```

`map` creates a new array from an existing array, where each element in the new array is created by passing the elements of the original array into the function passed to `map` (`function(n) { return n*2 }` in this case).

What the `map` function does is abstract away the process of explicitly iterating over the array, and lets us focus on *what* we want to happen. Note that the function we pass to map is pure; it doesn't have any side effects (change any external state), it just takes in a number and returns the number doubled.

From http://latentflip.com/imperative-vs-declarative/

PROCEDURAL PROGRAMMING

- Declarative programming simply *abstracts* away the details of *what* to do.

PROCEDURAL PROGRAMMING

- Declarative programming simply *abstracts* away the details of *what* to do.
- We will NOT do declarative programming.

PROCEDURAL PROGRAMMING

- Declarative programming simply *abstracts* away the details of *what* to do.
- We will NOT do declarative programming.
- We will do a type of Imperative programming, which is a further abstraction from a purely procedural approach such as C; namely, Object-Oriented (*i.e.*, C++ or Java).

It's Actually More Complicated

Paradigm	Description	Main traits	Related paradigm(s)	Critique	Examples
Imperative	Programs as statements that *directly* change computed state (datafields)	Direct assignments, common data structures, global variables		Edsger W. Dijkstra, Michael A. Jackson	C, C++, Java, PHP, Python, Ruby
Structured	A style of imperative programming with more logical program structure	Structograms, indentation, no or limited use of goto statements	Imperative		C, C++, Java, Python
Procedural	Derived from structured programming, based on the concept of modular programming or the *procedure call*	Local variables, sequence, selection, iteration, and modularization	Structured, imperative		C, C++, Lisp, PHP, Python
Functional	Treats computation as the evaluation of mathematical functions avoiding state and mutable data	Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects	Declarative		C++,[1] Clojure, Coffeescript,[2] Elixir, Erlang, F#, Haskell, Lisp, Python, Ruby, Scala, SequenceL, Standard ML
Event-driven including time-driven	Control flow is determined mainly by events, such as mouse clicks or interrupts including timer	Main loop, event handlers, asynchronous processes	Procedural, dataflow		JavaScript, ActionScript, Visual Basic, Elm
Object-oriented	Treats datafields as *objects* manipulated through predefined methods only	Objects, methods, message passing, information hiding, data abstraction, encapsulation, polymorphism, inheritance, serialization-marshalling	Procedural	Here and[3][4][5]	Common Lisp, C++, C#, Eiffel, Java, PHP, Python, Ruby, Scala
Declarative	Defines program logic, but not detailed control flow	Fourth-generation languages, spreadsheets, report program generators			SQL, regular expressions, CSS, Prolog, OWL, SPARQL
Automata-based programming	Treats programs as a model of a finite state machine or any other formal automata	State enumeration, control variable, state changes, isomorphism, state transition table	Imperative, event-driven		Abstract State Machine Language

https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms

Classification of Algorithms by Task

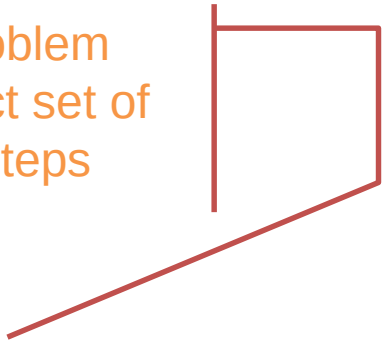
Searching versus Sorting (covered later).

Classification of Algorithms by Technique

Algorithmic versus Heuristic

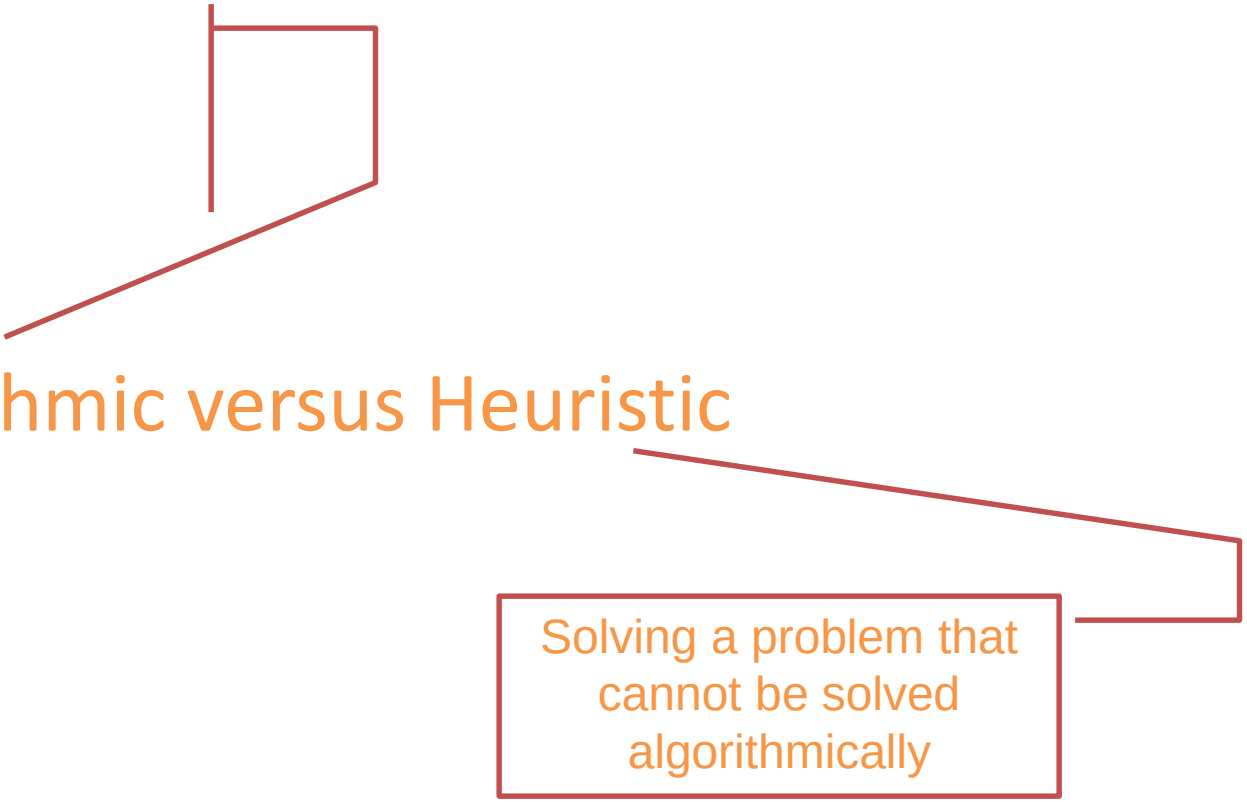
Classification of Algorithms by Technique

Solving a problem through a direct set of sequential steps

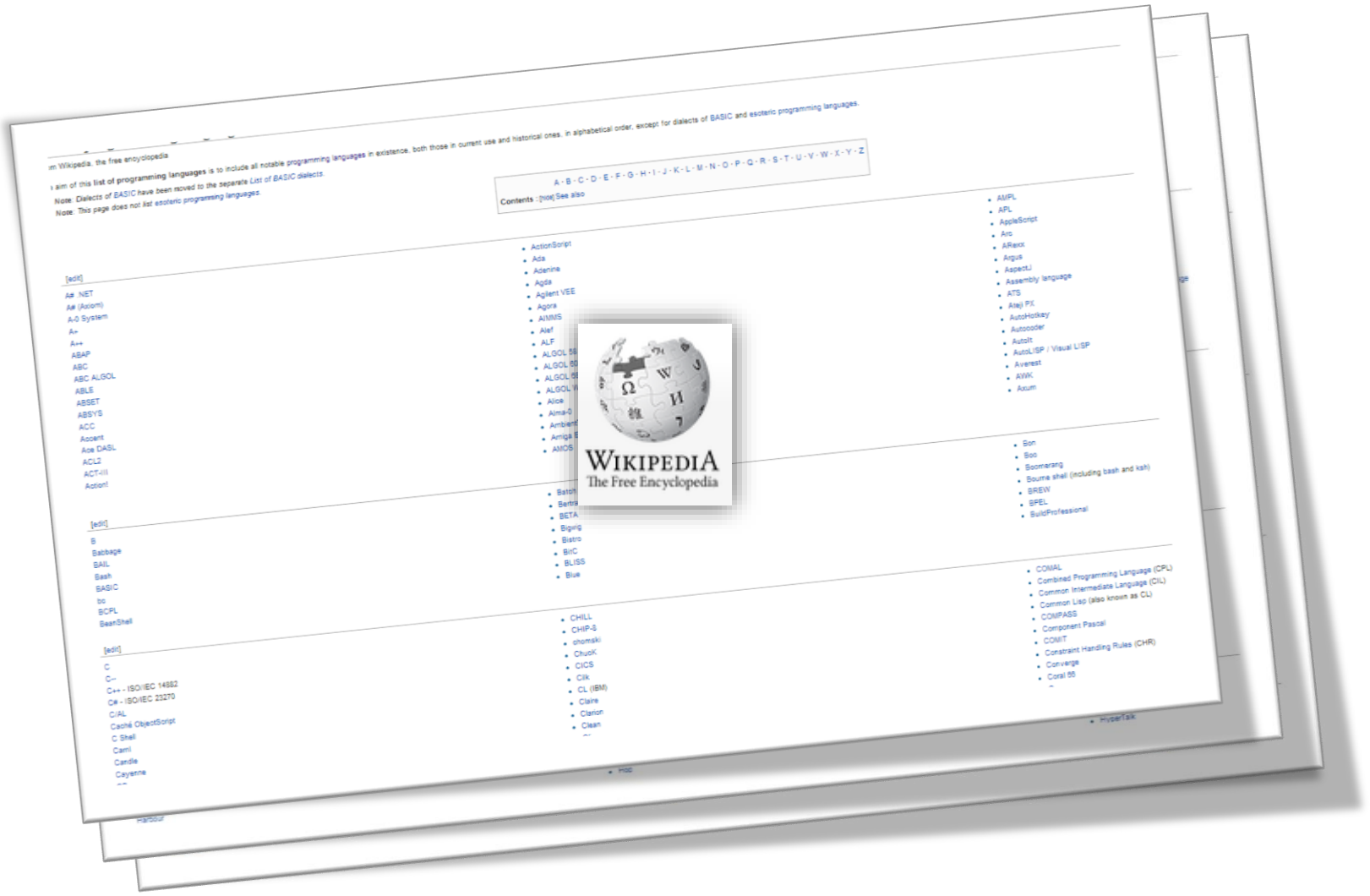


Algorithmic versus Heuristic

Classification of Algorithms by Technique



Different Kinds of Languages



Different Kinds of Languages

- **FACT**: Any programming language can solve any solvable problem (discussed later).

Different Kinds of Languages

- **FACT**: Any programming language can solve any solvable problem (discussed later).
- **FACT**: A specific programming language is designed for specific types of problems. For example,
  - **FORTRAN** is FORmula TRANslator (math),
  - **COBOL** was used for business (accounting),
  - **LISP** is used in AI research,
  - **JAVA** "Write once, run anywhere“ – very versitile.

Epilogue on Abstraction

Definition

ab·strac·tion

/abˈstrakSHən/

noun

1. the quality of dealing with ideas rather than events.
   "topics will vary in degrees of abstraction"
2. freedom from representational qualities in art.
   "geometric abstraction has been a mainstay in her work"

Abstraction

Abstraction in mathematics is the process of extracting the underlying essence of a mathematical concept, removing any dependence on real world objects with which it might originally have been connected, and generalizing it so that it has wider applications or matching among other abstract descriptions of equivalent phenomena. Two of the most highly abstract areas of modern mathematics are category theory and model theory.


Presentation
Terminated