

[illegible]

Tentative Syllabus

Week	Topic	Study Guide Topic
1 (Sept. 4 th)	Introduction to Course	Unit 1: What is programming about?
2 (11 th)	Introduction to Programming	
3 (18 th)	Programming in VB	Unit 2: Programming in Visual Basic
4 (29 th)	Events	Unit 3: Events
5 (Oct. 2 nd)	Representing and Storing Values	Unit 4: Representing and Storing Values
6 (19 th)	Subprograms	Unit 5: Subprograms
7 (20 th)	MIDTERM	Unit 6: More on Subprograms
8 (23 rd)	Decisions	Unit 7: Decisions
9 (30 th)	Iteration	Unit 8: Repetition
10 (Nov. 6 th)	Arrays	Unit 9: Representing Lists and Tables with Arrays
11 (13 th)	I/O	Unit 10: File Input and Output
12 (21 st)	Graphics	Unit 11: Graphs and Simulation
13 (27 th)	Review	

Review from Last Time

- Introduced the course (beginners).
- Defined Computing Science.
- Defined Algorithm.
- Provided a historical overview.
- Defined GUI and CLI user interfaces in their historic contexts.

GOALS



- **Fundamental definitions**
- **Very Brief historical background**
- **Fundamental principles**
- **Classification Schemes**

Definition: Algorithm

A recipe for performing a task such as constructing something, or solving a problem whose solution is unknown (*source unknown*).

Definition: Algorithm

A recipe for performing a task such as constructing something, or solving a problem whose solution is unknown (*source unknown*).

In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning. [Wikipedia](#)

Better Definition: Algorithm

A well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Schneider and Gersting, 2004

Better Definition: Algorithm

A well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Schneider and Gersting, 2004

QUESTION: What is the difference between a recipe and an algorithm?

NOTES

- Be sure that you understand the importance of each of the underlined terms and how their synthesis provides this complete definition for *algorithm*.

NOTES

- Be sure that you understand the importance of each of the underlined terms and how their synthesis provides this complete definition for *algorithm*.
- Remember that this is NOT a universally accepted definition (universal definitions are usually found in classical theories, such as the algebra of real numbers, logic, set theory, *etc.*

NOTES

- Be sure that you understand the importance of each of the underlined terms and how their synthesis provides this complete definition for *algorithm*.
- Remember that this is NOT a universally accepted definition (universal definitions are usually found in classical theories, such as the algebra of real numbers, logic, set theory, *etc.*
- A ***program*** is the implementation of an algorithm (usually on a computer) and is considered a ***formal language***.

NOTES

- Be sure that you understand the importance of each of the underlined terms and how their synthesis provides this complete definition for *algorithm*.
- Remember that this is NOT a universally accepted definition (universal definitions are usually found in classical theories, such as the algebra of real numbers, logic, set theory, *etc.*
- A ***program*** is the implementation of an algorithm (usually on a computer) and is considered a ***formal language***.
- **NOTE: Code** and ***program*** are often used synonymously.

Definition: Formal Language

In mathematics, computer science, and linguistics, a formal language is a set of strings of symbols that may be constrained by rules that are specific to it. [Wikipedia](#)

Definition: Formal Language

In mathematics, computer science, and linguistics, a formal language is a set of strings of symbols that may be constrained by rules that are specific to it. [Wikipedia](#)

Related topics

In formal language theory, a grammar (when the context is not given, often called a **formal grammar** for clarity) is a set of **production rules** for strings in a formal language. [Wikipedia](#)

Explore: [Formal grammar](#)

Natural language is distinguished from constructed languages and formal languages such as computer-programming languages or the “languages” used in the study of formal logic, especially mathematical logic. [Wikipedia](#)

Definition: Formal Language

Natural language

In neuropsychology, linguistics and the philosophy of language, a natural language or ordinary language is any language which arises, unpremeditated, in the brains of human beings. [Wikipedia](#)

Pseudocode

- A computer language is a formal language.

Pseudocode

- A computer language is a formal language.
- English is a natural language.

Pseudocode

- A computer language is a formal language.
- English is a natural language.
- Pseudocode is in between the two, describing what we want a program to do in English, before translating the solution into a programming language.

Pseudocode

- For example, for making a cup of tea:

Pseudocode

- For example, for making a cup of tea:

```
Organise everything together;  
Plug in kettle;  
Put teabag in cup;  
Put water into kettle;  
Wait for kettle to boil;  
Add water to cup;  
Remove teabag with spoon/fork;  
Add milk and/or sugar;  
Serve;
```

Think of pseudocode as the recipe for a program (it lacks *formal* syntax).

Class Exercise

Task

Compute the Root mean square of the numbers 1..10.

Class Exercise

The RMS is calculated as the mean of the squares of the numbers, square-rooted:

$$x_{\text{rms}} = \sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}}.$$

Class Exercise – An Answer

```
ms = 0
for i = 1 ... N
    ms = ms + y[i]^2
ms = ms / N
rms = sqrt(ms)
```

i.e. the square root of the mean of the squared values of elements of `y`.

VBA Code for this Problem

```
Function rms(iLow As Integer, iHigh As Integer)
    Dim i As Integer
    If iLow > iHigh Then
        i = iLow
        iLow = iHigh
        iHigh = i
    End If
    For i = iLow To iHigh
        rms = rms + i ^ 2
    Next i
    rms = Sqr(rms / (iHigh - iLow + 1))
End Function

Sub foo()
    Debug.Print rms(1, 10)
End Sub
```


Pseudocode with Syntax-Style

Fortran style pseudo code	Pascal style pseudo code	C style pseudo code:	Basic style pseudo code
<pre>program fizzbuzz Do i = 1 to 100 set print_number to true If i is divisible by 3 print "Fizz" set print_number to false If i is divisible by 5 print "Buzz" set print_number to false If print_number, print i print a newline end do</pre>	<pre>procedure fizzbuzz For i := 1 to 100 do set print_number to true; If i is divisible by 3 then print "Fizz"; set print_number to false; If i is divisible by 5 then print "Buzz"; set print_number to false; If print_number, print i; print a newline; end</pre>	<pre>void function fizzbuzz { for (i = 1; i <= 100; i++) { set print_number to true; If i is divisible by 3 print "Fizz"; set print_number to false; If i is divisible by 5 print "Buzz"; set print_number to false; If print_number, print i; print a newline; } }</pre>	<pre>Sub fizzbuzz() For i = 1 to 100 print_number = True If i is divisible by 3 Then Print "Fizz" print_number = False End If If i is divisible by 5 Then Print "Buzz" print_number = False End If If print_number = True Then print i Print a newline Next i End Sub</pre>

Examples of pseudocode for a mathematical game ([fizz buzz](#)) with the *syntax-style* of four different programming languages.

Pseudocode

- Pseudocode can be highly structured (looking like some programming language).

Pseudocode

- Pseudocode can be highly structured (looking like some programming language).
- There are standards, however, and an excellent summary is given here:
 - <http://faculty.ccri.edu/mkelly/COMI1150/PseudocodeBasics.pdf>
 - http://www.cosc.canterbury.ac.nz/tim.bell/dt/Tutorial_Pseudocode.pdf

Pseudocode

- Pseudocode can be highly structured (looking like some programming language).
- There are standards, however, and an excellent summary is given by:
 - <http://faculty.ccri.edu/mkelly/COMI1150/PseudocodeBasics.pdf>
- In particular, note the use of indentation for control structures (extended to programs).

Final Note on Pseudocode

- Try not to make pseudocode too formal.

Final Note on Pseudocode


- Try not to make pseudocode too formal.
- Perhaps, this is as formal as necessary (Pascal-like style):

ALGORITHM 1 Finding the Maximum Element in a Finite Sequence.

```
procedure max( $a_1, a_2, \dots, a_n$ : integers)
  max :=  $a_1$ 
  for  $i := 2$  to  $n$ 
    if  $max < a_i$  then  $max := a_i$ 
  return  $max$ {max is the largest element}
```

Flow Charts

flow chart

/ˈflō ˌCHärt/ 

noun

noun: flowchart

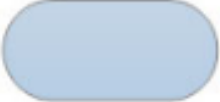


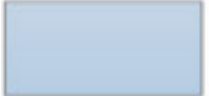

a diagram of the sequence of movements or actions of people or things involved in a complex system or activity.

- a graphical representation of a computer program in relation to its sequence of functions (as distinct from the data it processes).

Flow chart Cheat Sheet:

<http://richbodo.pbworks.com/w/file/fetch/104325789/Gliffy-flow-chart-cheat-sheet.pdf>

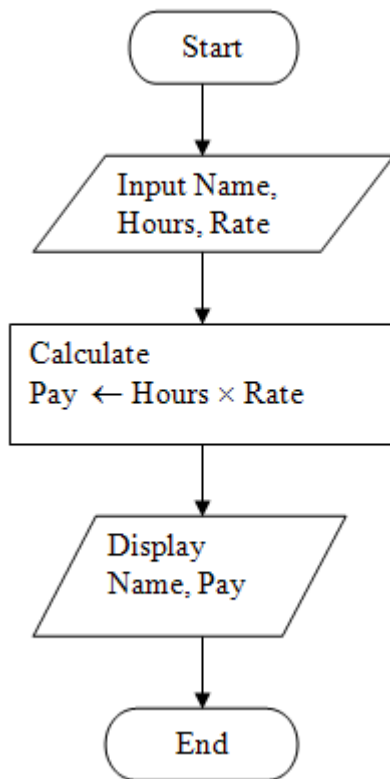
Flow Charts

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Exercise: Draw the Associated Flowchart

1. Input the three values into the variables Name, Hours, Rate.
2. Calculate $\text{Pay} = \text{Hours} * \text{Rate}$.
3. Display Name and Pay.

Exercise: Draw the Associated Flowchart



1. Input the three values into the variables Name, Hours, Rate.
2. Calculate $\text{Pay} = \text{Hours} * \text{Rate}$.
3. Display Name and Pay.

Exercise: Add 10 and 20

To solve this problem we will take a variable sum and set it to zero. Then we will take the two numbers 10 and 20 as input. Next we will add both the numbers and save the result in the variable sum i.e., $\text{sum} = 10 + 20$. Finally, we will print the value stored in the variable sum.

Exercise: Add 10 and 20

To solve this problem we will take a variable sum and set it to zero. Then we will take the two numbers 10 and 20 as input. Next we will add both the numbers and save the result in the variable sum i.e., $\text{sum} = 10 + 20$. Finally, we will print the value stored in the variable sum.

Algorithm (in simple English)

- Initialize sum = 0 (PROCESS)
- Enter the numbers (I/O)
- Add them and store the result in sum (PROCESS)
- Print sum (I/O)

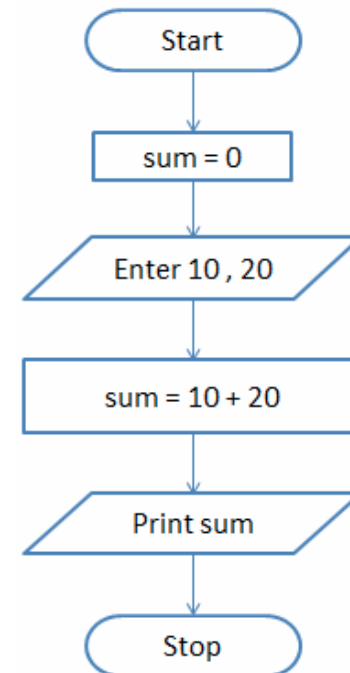
Exercise: Add 10 and 20

To solve this problem we will take a variable sum and set it to zero. Then we will take the two numbers 10 and 20 as input. Next we will add both the numbers and save the result in the variable sum i.e., $\text{sum} = 10 + 20$. Finally, we will print the value stored in the variable sum.

Algorithm (in simple English)

- Initialize $\text{sum} = 0$ (PROCESS)
- Enter the numbers (I/O)
- Add them and store the result in sum (PROCESS)
- Print sum (I/O)

Flowchart



Exercise: Sum of Five Numbers

In this question we are asked to find the sum of 5 numbers. So, we will take two variables - sum and count and set both of them to zero. The sum variable will store the result while the count variable will keep track of how many numbers we have read.

To solve this problem we will use the concept of loop. In loop or iterative operation, we execute some steps repeatedly as long as the given condition is TRUE. In this case we will keep reading the input till we have read 5 numbers.

So, we first initialize sum and count to zero. Then we will take the input and store it in a variable n. Next we will add the value stored in n to sum and save the answer in sum.

i.e., $\text{sum} = \text{sum} + n$

Then we will increment count by 1 and check if count is less than 5. If this condition is TRUE then we will take another input. If the condition is FALSE then we will print the value stored in variable sum.

Exercise: Add 10 and 20

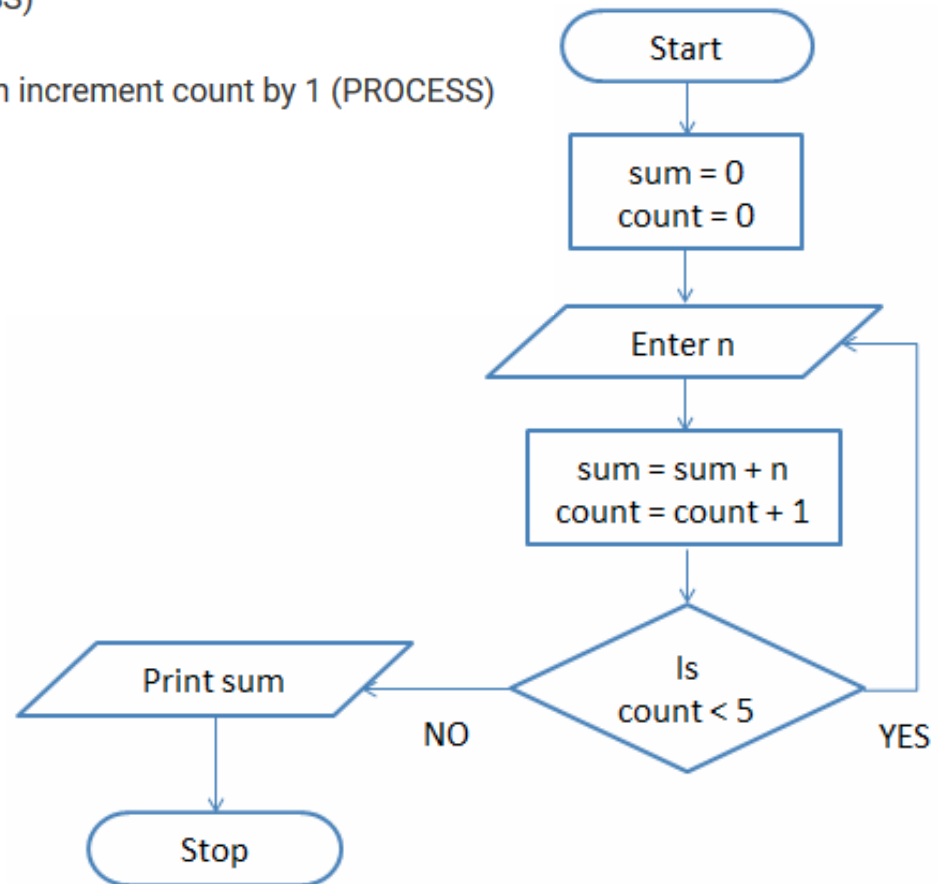
Algorithm (in simple English)

1. Initialize sum = 0 and count = 0 (PROCESS)
2. Enter n (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
5. if YES go to step 2
else
Print sum (I/O)

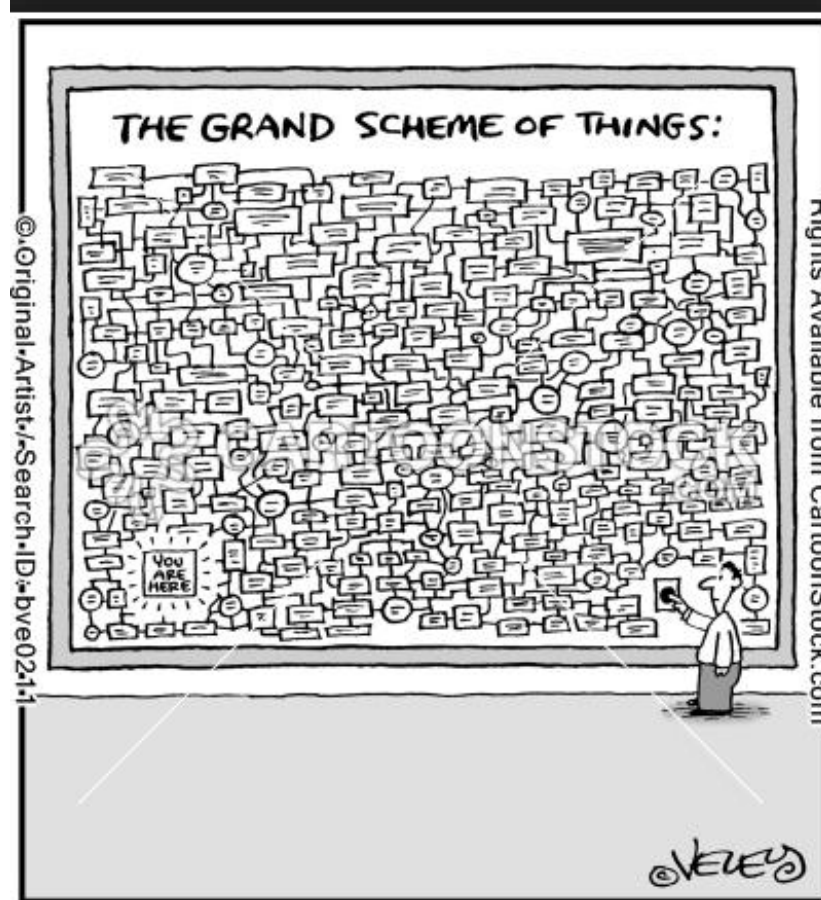
Exercise: Add 10 and 20

Algorithm (in simple English)

1. Initialize sum = 0 and count = 0 (PROCESS)
2. Enter n (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
5. if YES go to step 2
else
Print sum (I/O)



Flow Charts



See the following website regarding tips for creating good flowcharts:
<http://creately.com/blog/diagrams/part-1-15-mistakes-you-would-unintentionally-make-with-flowcharts/>

Characteristics of an Algorithm

An algorithm has two general characteristics:

- ***Representation:*** A concrete, symbolic encoding of information, e.g. numbers, words, names.
- ***Transformation:*** The steps (recipe, *program, algorithm*) used to calculate a specific result.

Characteristics of an Algorithm

Two Formal Characteristics of Language

- **Syntax** (grammar) specifies the *form* of a language by describing the possible combination of symbols which, in turn, can be text-based or graphical in nature.

Two Formal Characteristics of Language

- **Syntax** (grammar) specifies the *form* of a language by describing the possible combination of symbols which, in turn, can be text-based or graphical in nature.
- **Semantics** specifies the meaning conveyed by a language (see Wikipedia, Programming Languages).

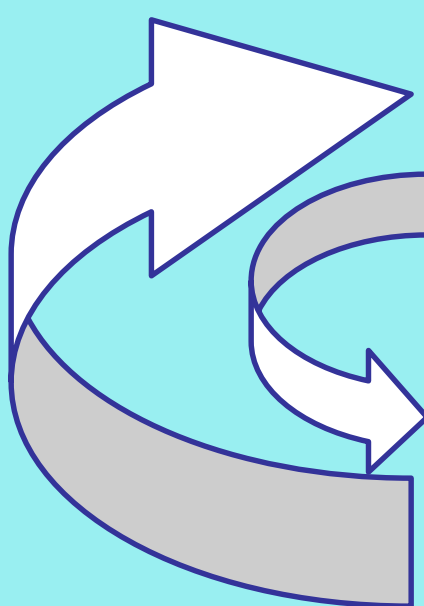
The Flow of Logic

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.”

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 repeat or require decisions 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 $\frac{1}{4}$ 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.

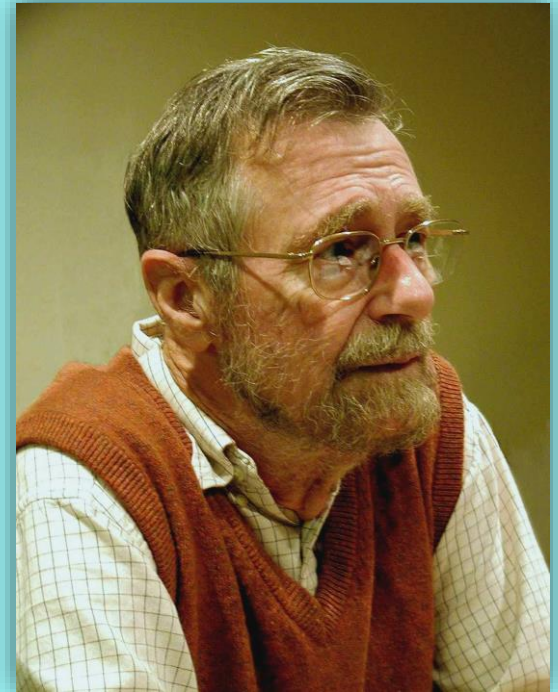
A recipe is a sequence of instructions

- Making pancakes:
 - Gather ingredients.
 - **Beat** egg **until** homogeneous.
 - **Mix** in water, oil and cereal **until** homogeneous.
 - Pour resulting batter onto hot, oiled pan using $\frac{1}{4}$ 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.

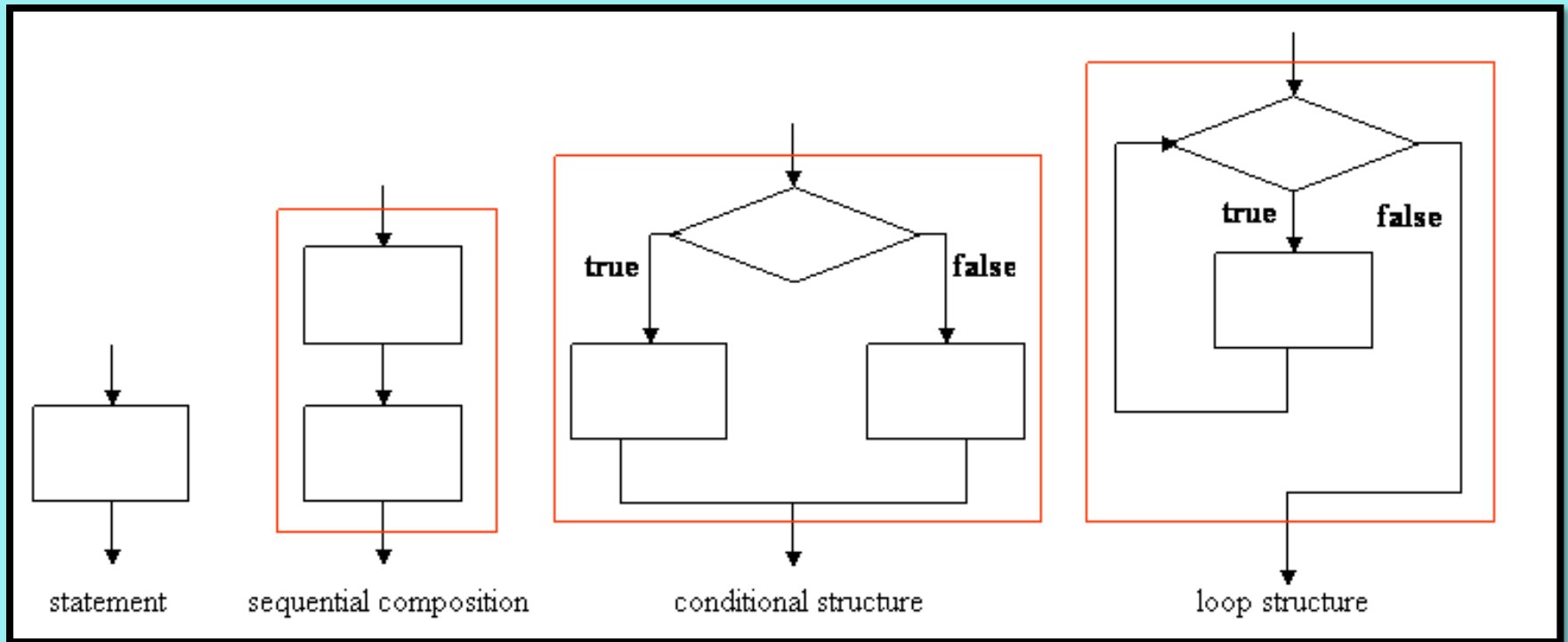
Edsger W. Dijkstra

1930–2002

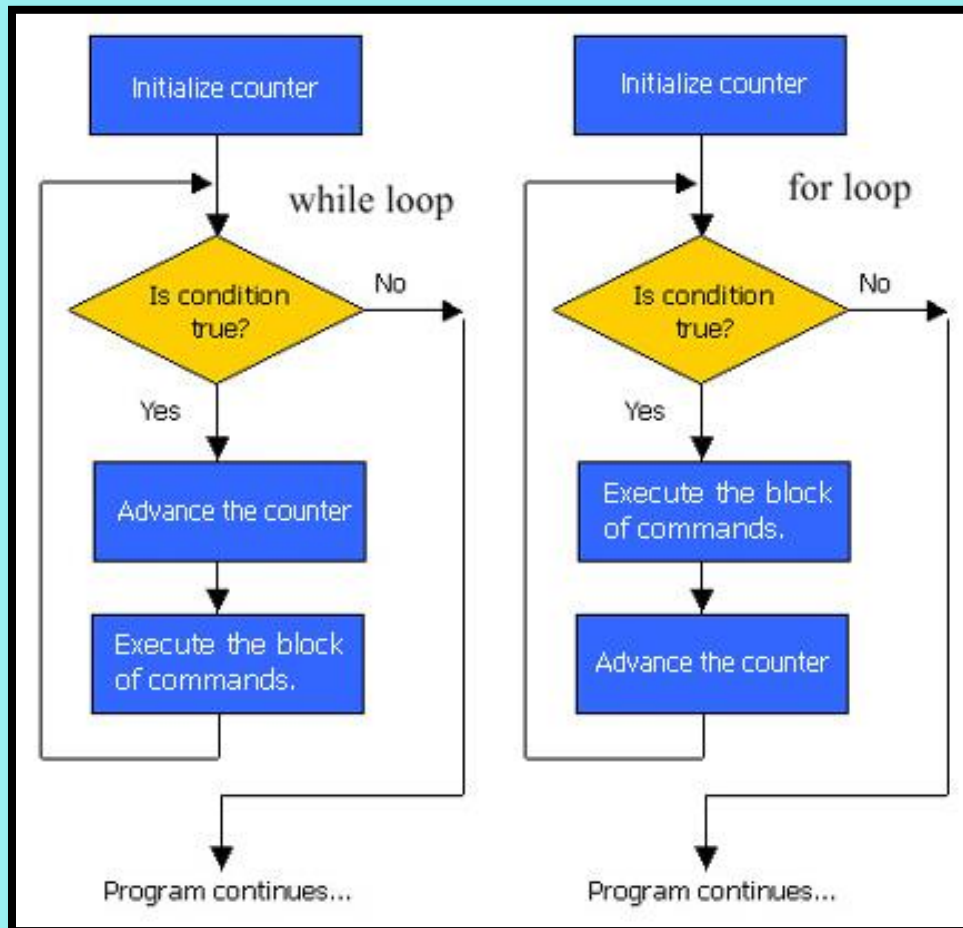
- Dijkstra (and others) in the late 1960's recognized that the *flow of logic* in code can be organized according to three primary **CONTROL STRUCTURES**
 1. Sequence
 2. Iteration (repeating a procedure)
 3. Decision



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 “Distance” from Hardware

- Computers are binary and digital at their core.

Classification by “Distance” from Hardware

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

Classification by “Distance” 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 “Distance” 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 “Distance” 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 VB.

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
```

Machine Language

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

```
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

1GL, 2GL, 3GL

M
o
r
e
u
s
e
r
f
r
i
e
n
d
l
y



First generation Machine language

```
11110010 01110011 1101 001000010000 0111 000000101011
11110010 01110011 1101 001000011000 0111 000000101111
11111100 01010010 1101 001000010010 1101 001000011101
11110000 01000101 1101 001000010011 0000 000000111110
11110011 01000011 0111 000001010000 1101 001000010100
10010110 11110000 0111 000001010100
```



Second generation Assembly language

```
PACK 210(8,13),02B(4,7)
PACK 218(8,13),02F(4,7)
MP 212(6,13),21D(3,13)
SRP 213(5,13),03E(0),5
UNPK 050(5,7),214(4,13)
OI 054(7),X F0
```



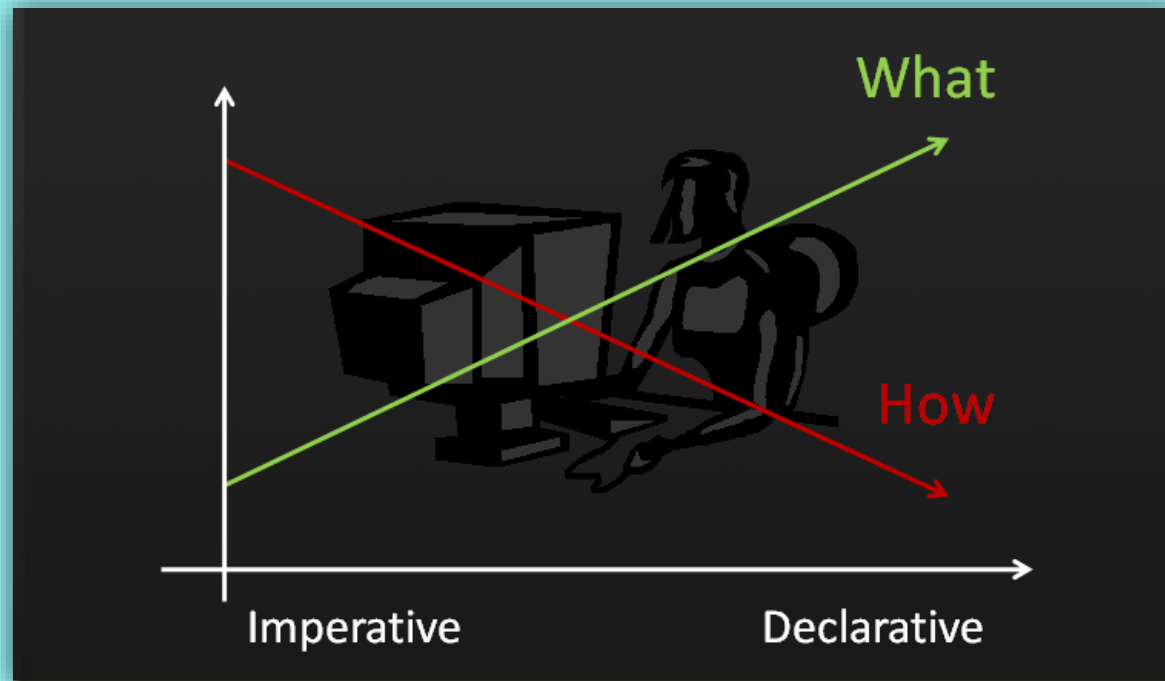
Third generation COBOL

```
MULTIPLY HOURS-WORKED BY PAY-RATE GIVING GROSS-PAY ROUNDED
```

Classification of Algorithms by Implementation

**Programming Paradigms
(Fundamental Abstraction)**

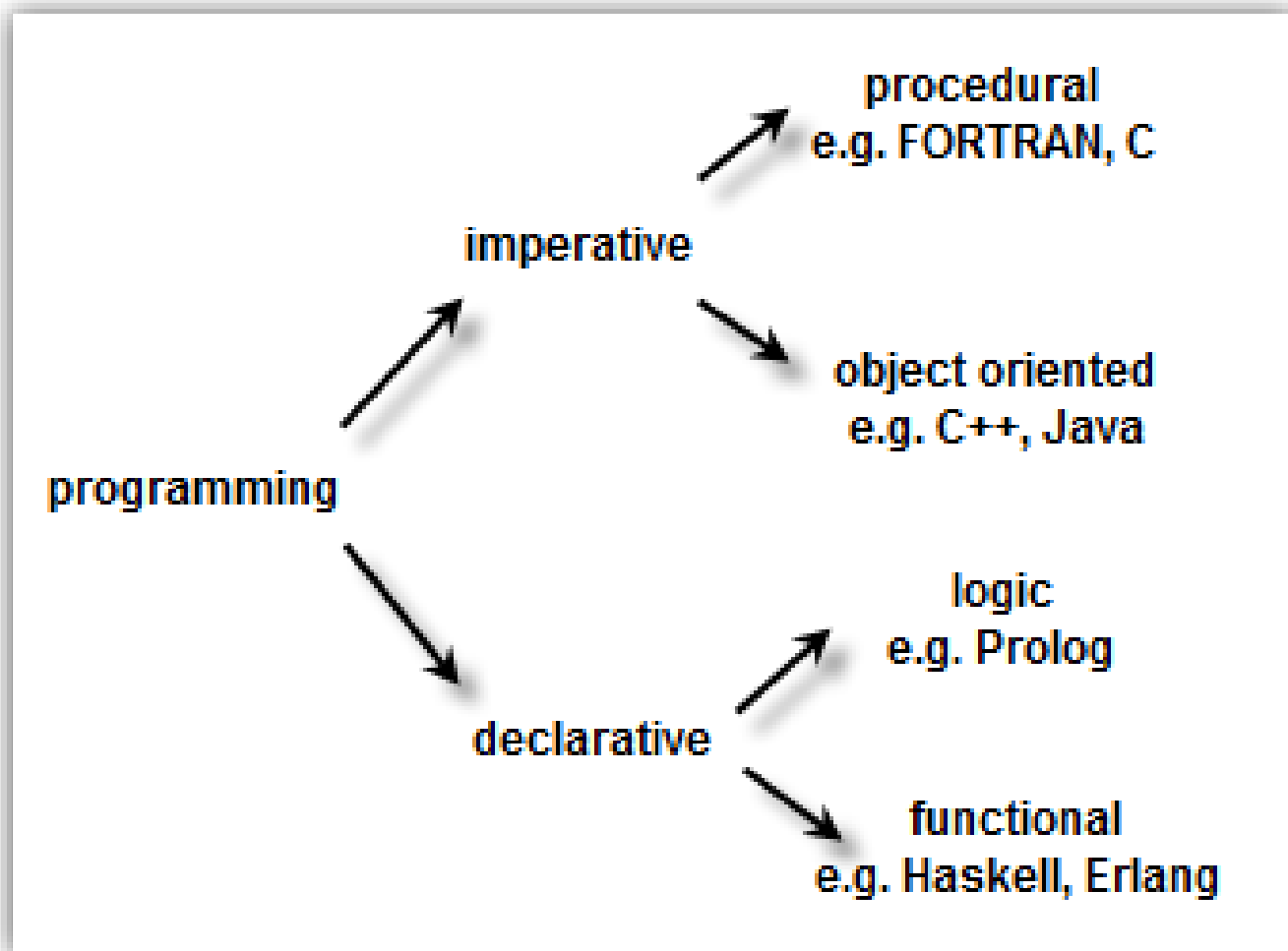
Implementation Classification



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



Classifying Languages by Implementation

Declarative programming is a programming paradigm ... that expresses the logic of a computation without describing its control flow.

Imperative programming is a programming paradigm that uses statements that change a program's state.

Classifying Languages by Implementation

Declarative programming is a programming paradigm ... that expresses the logic of a computation without describing its control flow.

Imperative programming is a programming paradigm that uses statements that change a program's state.

- *Declarative Programming* is like asking your friend to draw a landscape. *You don't care how they draw it, that's up to them.*
- *Imperative Programming* is like your friend listening to Bob Ross tell them how to paint a landscape. While good ole Bob Ross isn't exactly commanding, he is giving them *step by step directions* to get the desired result.

Classifying Languages by Implementation

Declarative programming is a programming paradigm ... that expresses the logic of a computation without describing its control flow.

Imperative programming is a programming paradigm that uses statements that change a program's state.

- *Declarative Programming* is like asking your friend to draw a landscape. *You don't care how they draw it, that's up to them.*
WHAT TO DO
- *Imperative Programming* is like your friend listening to Bob Ross tell them how to paint a landscape. While good ole Bob Ross isn't exactly commanding, he is giving them *step by step directions* to get the desired result.
HOW TO DO IT

PROCEDURAL PROGRAMMING

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

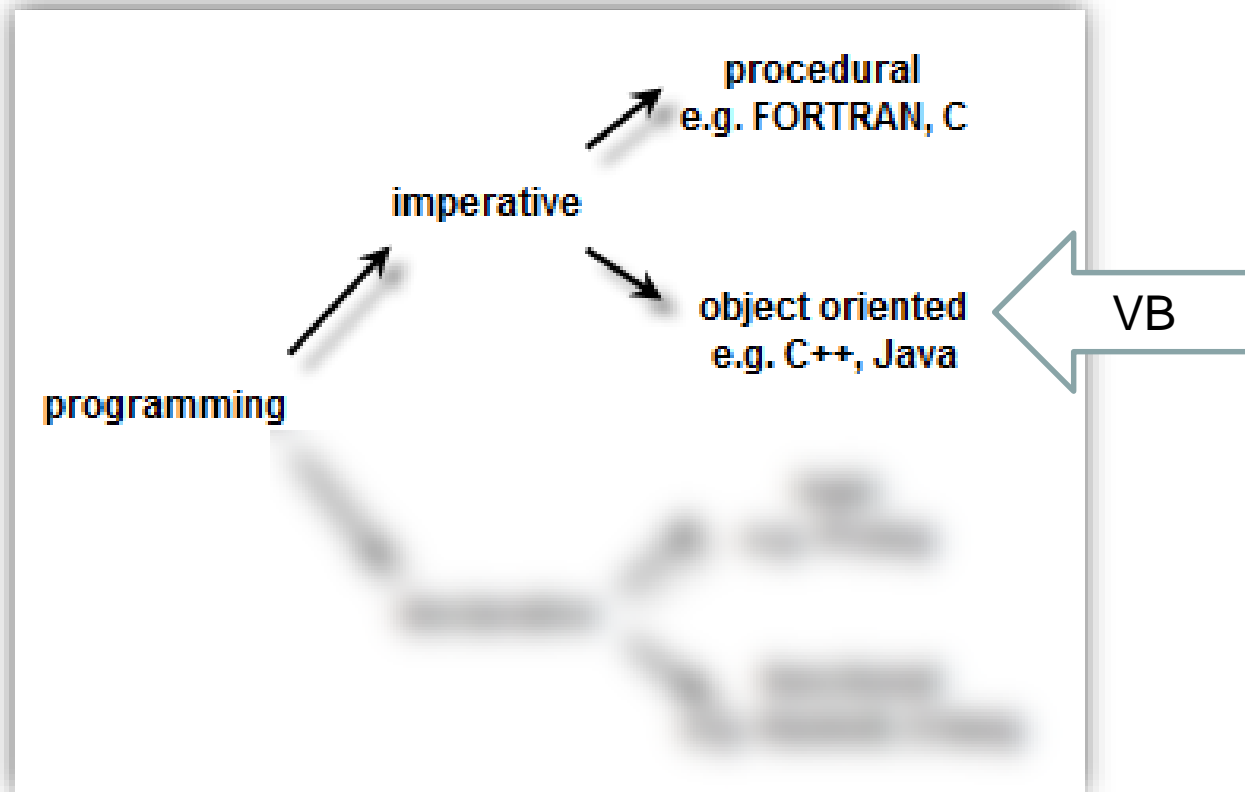
PROCEDURAL PROGRAMMING

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

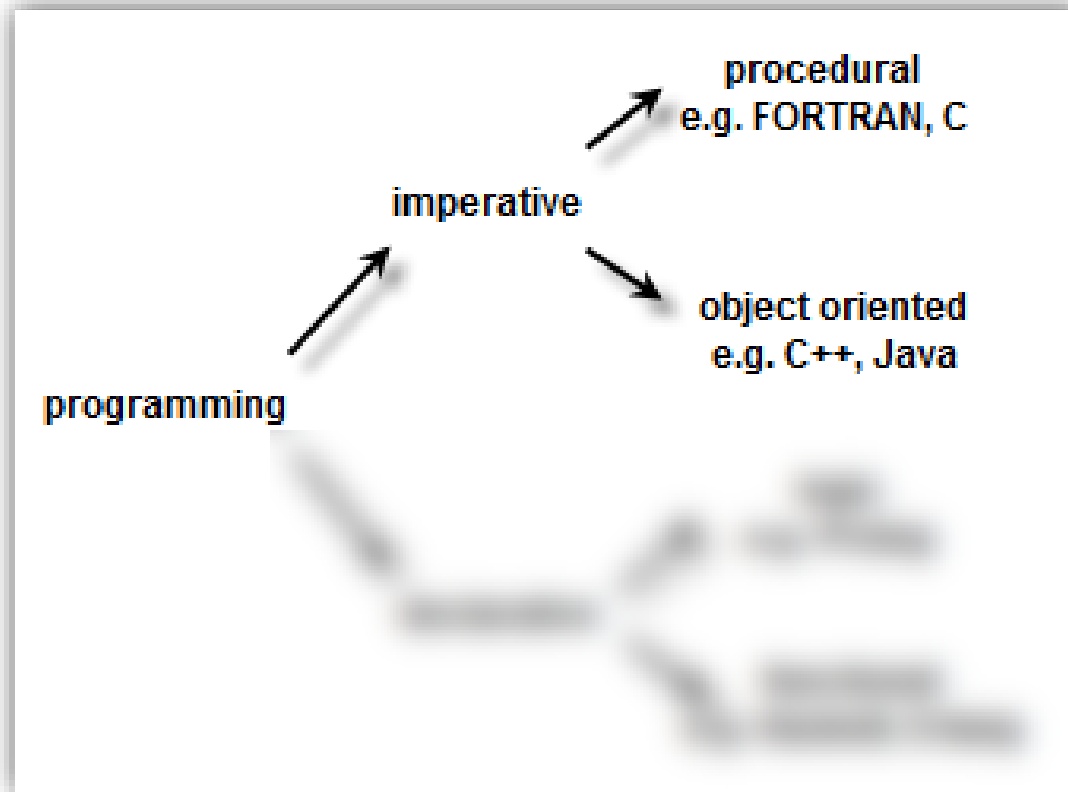
PROCEDURAL PROGRAMMING

- Declarative programming simply *abstracts* away the details of *what* to do.
- We will NOT do declarative programming.
We will do the simplest type of programming, which is **OO** (a subset of imperative programming where you use procedures, such as functions, to describe the commands the computer should perform).

Classifying “Processing”



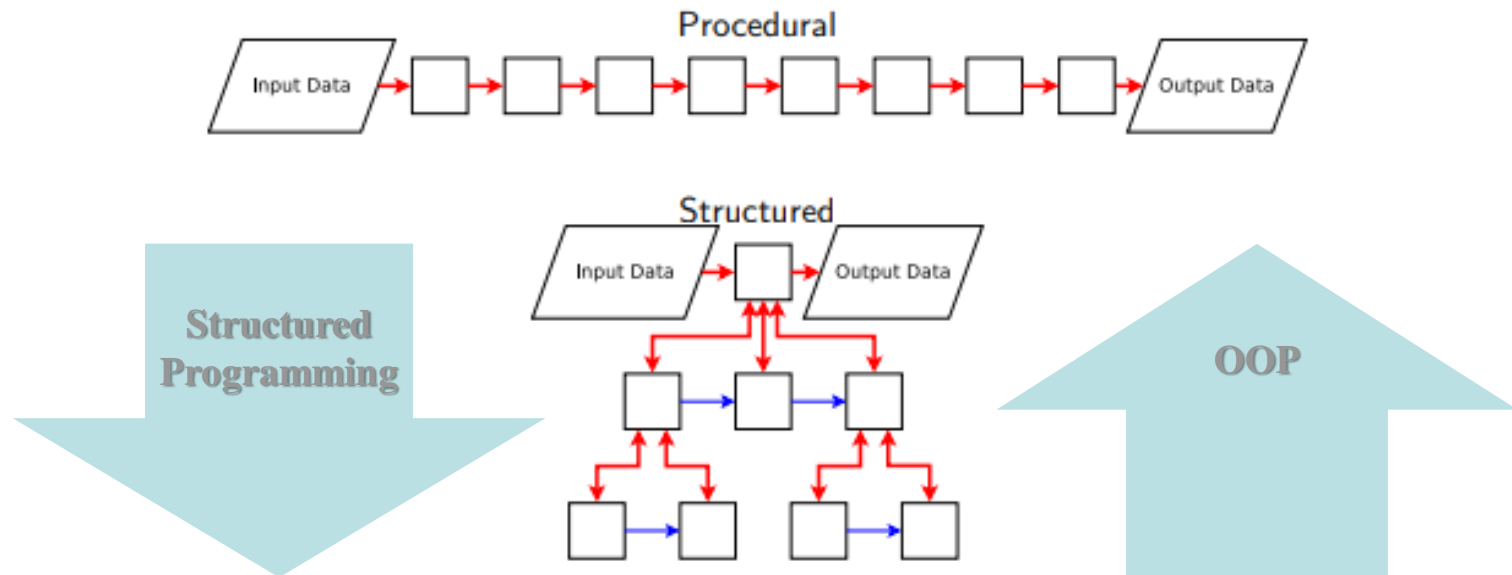
Classifying Languages by Implementation



Top-down

Bottom-up

The OO Paradigm Versus the Structured Paradigm



Classification of Algorithms by Task

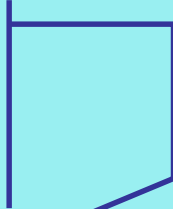
Searching versus Sorting

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

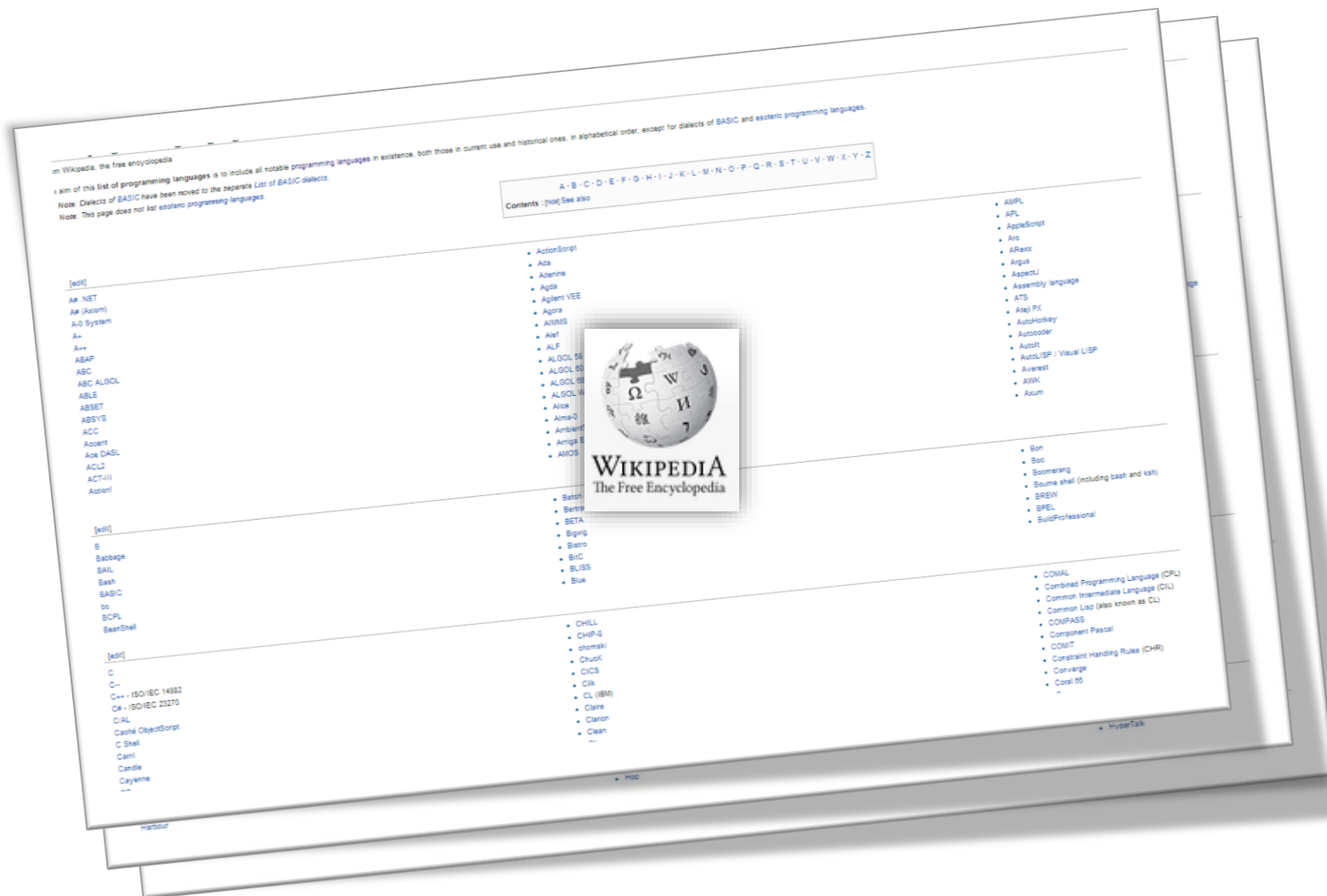
Classification of Algorithms by Technique

Solving a problem through a direct set of sequential steps

Algorithmic versus Heuristic

Solving a problem that cannot be solved algorithmically

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,
 - **PYTHON** was meant for education but has evolved into a more powerful language.

Hello World in Several Languages

A "Hello, world!" program is a computer program that outputs "Hello, World!" on a display device. It is typically one of the simplest programs possible in almost all computer languages. As such it can be used to quickly compare syntax differences between various programming languages. It is also used to verify that a language or system is operating correctly. The following is a list of "Hello, world" programs in 28 of the most commonly used programming languages.

1. Backbone.js

2. Bash

```
echo "Hello World"
```

3. Basic

```
PRINT "Hello, world!"
```


Hello World in Several Languages

A "Hello, world!" program is a computer program that outputs "Hello, World!" on a display device. It is typically one of the simplest programs possible in almost all computer languages. As such it can be used to quickly compare syntax differences between various programming languages. It is also used to verify that a language or system is operating correctly. The following is a list of "Hello, world" programs in 28 of the most commonly used programming languages.

4. C

1. Backbone.js

```
#include

int main(void)
{
    puts("Hello, world!");
}
```

2. Bash

```
echo "Hello World"
```

3. Basic

```
PRINT "Hello, world!"
```

5. C++

```
#include

int main()
{
    std::cout << "Hello, world!";
    return 0;
}
```

Hello World in Several Languages

A "Hello, world!" program is a computer program that outputs "Hello, World!" on a display device. It is typically one of the simplest programs possible in almost all computer languages. As such it can be used to quickly compare syntax differences between various programming languages. It is also used to verify that a language or system is operating correctly. The following is a list of "Hello, world" programs in 28 of the most commonly used programming languages.

1. Backbone.js

2. Bash

```
echo "Hello World"
```

3. Basic

```
PRINT "Hello, world!"
```

4. C

```
#include

int main(void)
{
    puts("Hello, world!");
}
```

5. C++

```
#include

int main()
{
    std::cout << "Hello, world!";
    return 0;
}
```

6. C#

```
using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

Hello World in Several Languages

A "Hello, world!" program is a computer program that outputs "Hello, World!" on a display device. It is typically one of the simplest programs possible in almost all computer languages. As such it can be used to quickly compare syntax differences between various programming languages. It is also used to verify that a language or system is operating correctly. The following is a list of "Hello, world" programs in 28 of the most commonly used programming languages.

1. Backbone.js

2. Bash

```
echo "Hello World"
```

3. Basic

```
PRINT "Hello, world!"
```

4. C

```
#include

int main(void)
{
    puts("Hello, world!");
}
```

5. C++

```
#include

int main()
{
    std::cout << "Hello, world!";
};

return 0;
}
```

6. C#

```
using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

27. Visual Basic .NET

```
Module Module1
    Sub Main()
        Console.WriteLine("Hello, world!")
    End Sub
End Module
```

Hello World in VB.Net

This is the world's favorite programming example, translated to Visual Basic .NET:

```
Imports System

Public Module Hello
    Public Sub Main( )
        Console.WriteLine("hello, world")
    End Sub
End Module
```

This version of *hello, world* is a *console application* -- it displays its output in a Windows command-prompt window. To compile this program, enter it using any text editor, such as Windows's Notepad, save it in a file whose name ends with *.vb*, such as *Hello.vb*, and compile it from the Windows command line with this command:

Watch any Hello World video...

VB Versus VBA

VBA stands for Visual Basic For Applications **and** its a Visual Basic implementation intended to be used in the Office Suite. The difference between them is that **VBA** is embedded inside Office documents (its an Office feature). **VB** is the ide/language for developing applications. Jun 14, 2009

Difference between Visual Basic 6.0 and VBA - Stack Overflow

<https://stackoverflow.com/questions/.../difference-between-visual-basic-6-0-and-vba>

IMPORTANT FINAL NOTE

COMMENTING CODE

```
city=raw_input("Enter a city: ")
while city[-1]!=" ":
    city = city[:-1]
temp=raw_input("Enter a temperature in Farenheit: ")
temp = float(temp)
temp = (temp - 32.0)*(100.0/180.0)
temp = round(temp,3)
temp = str(temp)
print "In "+city+" it is "+temp+" degrees Celcius!"
```

No comments results in grading penalties.

THE SAME CODE PROPERLY COMMENTED

```
#Alyssa P. Hacker
#fah_to_celsius.py

#collect a city name from user
city=raw_input("Enter a city: ")

#truncate whitespace
while city[-1]==" ":
    city = city[:-1]

#collect a temp from user
temp=raw_input("Enter a temperature in Farenheit: ")

#convert string to float
temp = float(temp)

#convert Farenheit temp to Celsius temp
temp = (temp - 32.0)*(100.0/180.0)

#truncate to 3 decimal places
temp = round(temp,3)

#recast as string so we can concatenate
temp = str(temp)

#print result!
print "In "+city+" it is "+temp+" degrees Celcius!"
```


The Alleged First Computer Programmer

Lady Lovelace



END PRESENTATION

