Introduction to Computing

Paradigms of Programming

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

August, 2024

Finding the maximum

Given three distinct integers as input, find out the maximum of them and show the same as output.

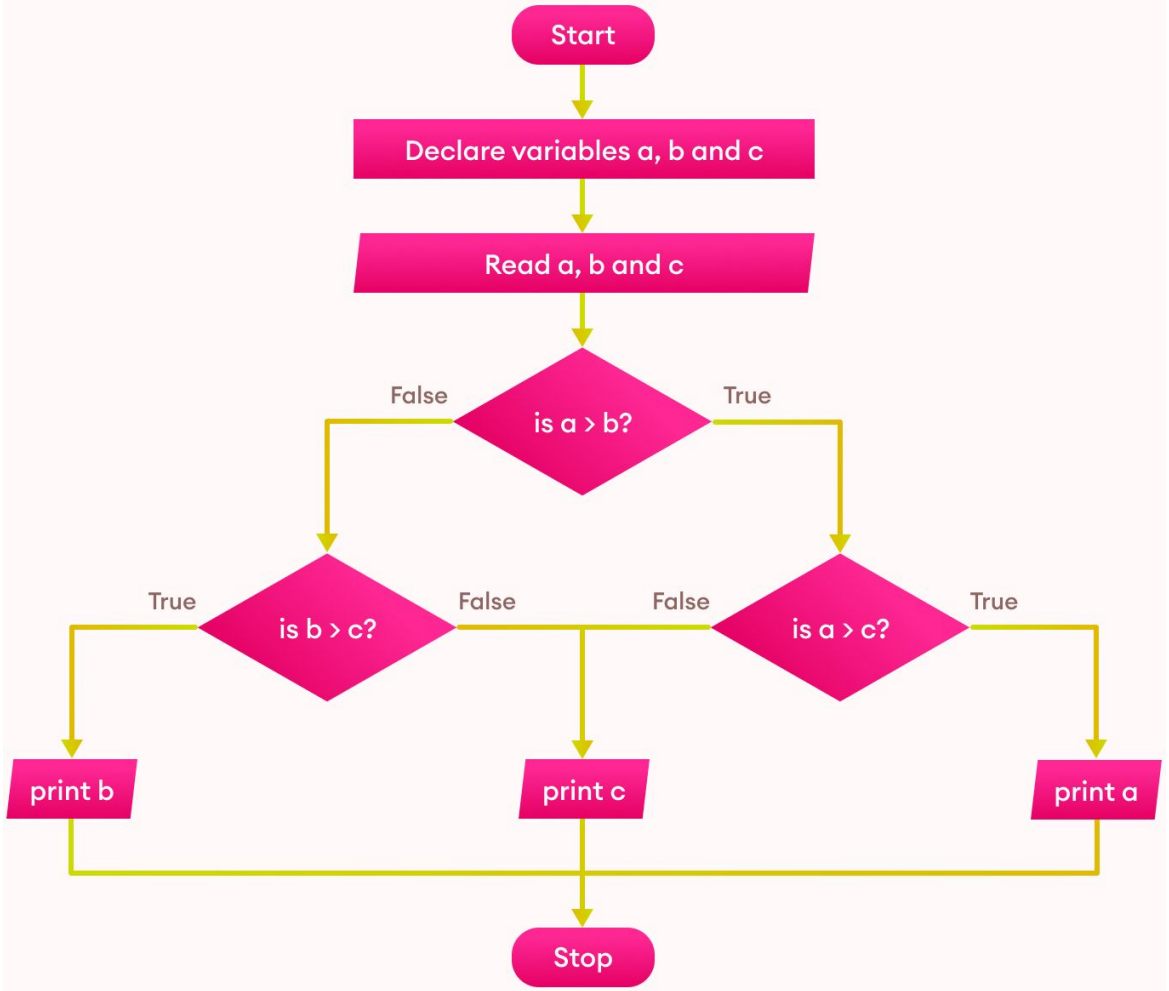
Choosing the maximum – A naive approach

```
Inputs: a, b, c  // All are distinct values
if a > b and a > c then do // 2 comparisons
        Output a
end if
if b > a and b > c then do // 2 comparisons
        Output b
end if
if c > a and c > b then do // 2 comparisons
        Output c
end if
```

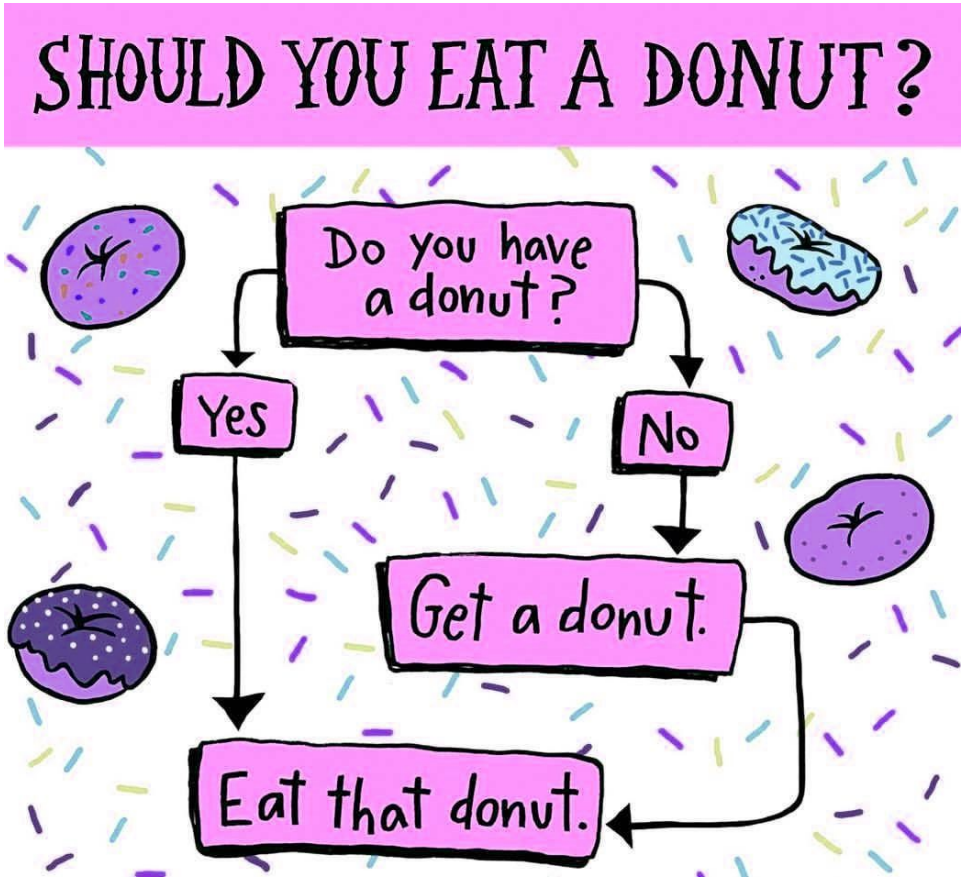
Finding the maximum – A better approach

```
Inputs: a, b, c // All are distinct values
if a > b then do // 1 comparison
    if a > c then do // 1 comparison
        Output a
    end if
    else do
        Output c
    end else
end if
else do
    if b > c then do // 1 comparison
        Output b
    end if
    else do
        Output c
    end else
end if
```

Finding the maximum – Using flowcharts



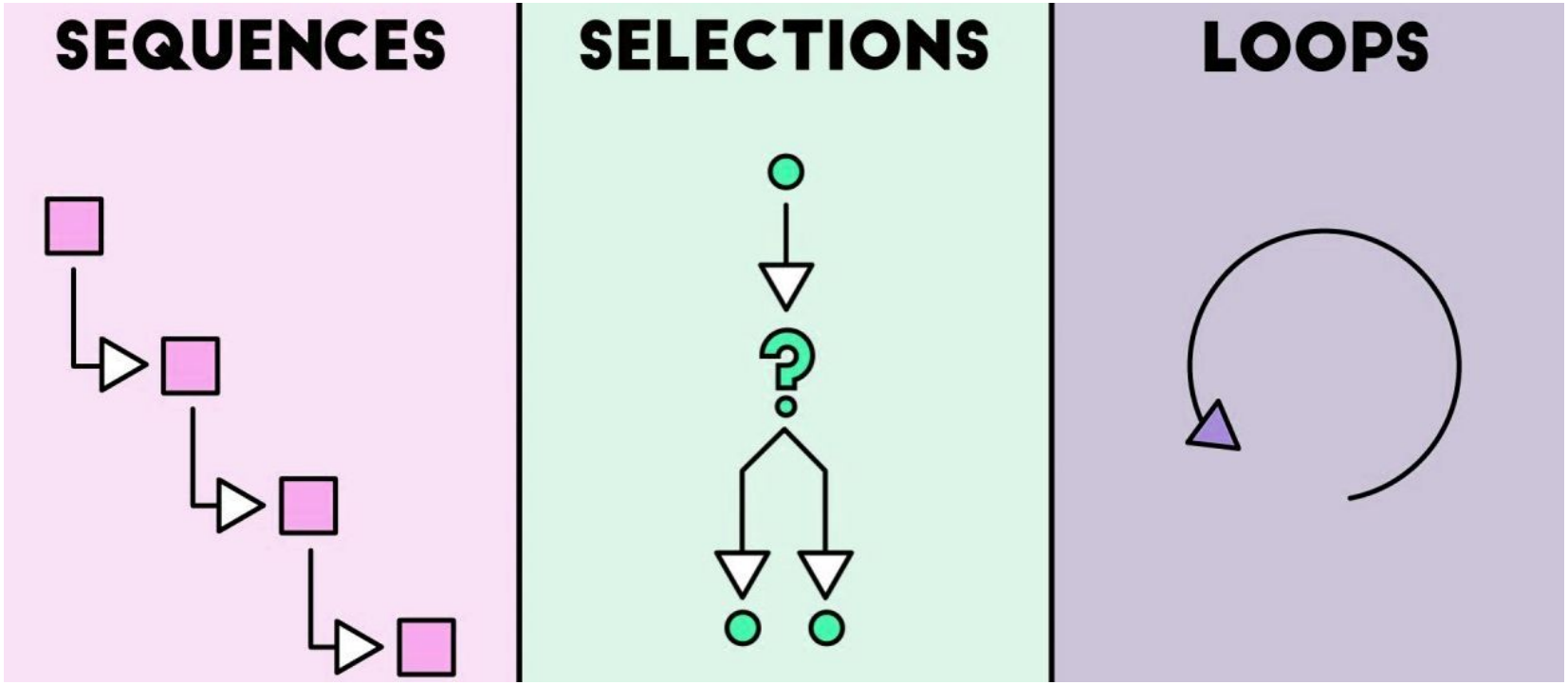
Decision making is everywhere



Basic symbols in a flowchart

Symbol	Name
	Start/end
	Arrows
	Input/Output
	Process
	Decision

Types of control flows



Illustrating the Euclid's algorithm

Given a pair of integers as input, find out their greatest common divisor (GCD) and show the same as output.

```
function gcd(a, b){
  while b != 0{
    t <- b
    b <- a % b
    a <- t
  }
  return a
}
```

Non-recursive (with Modular Division)

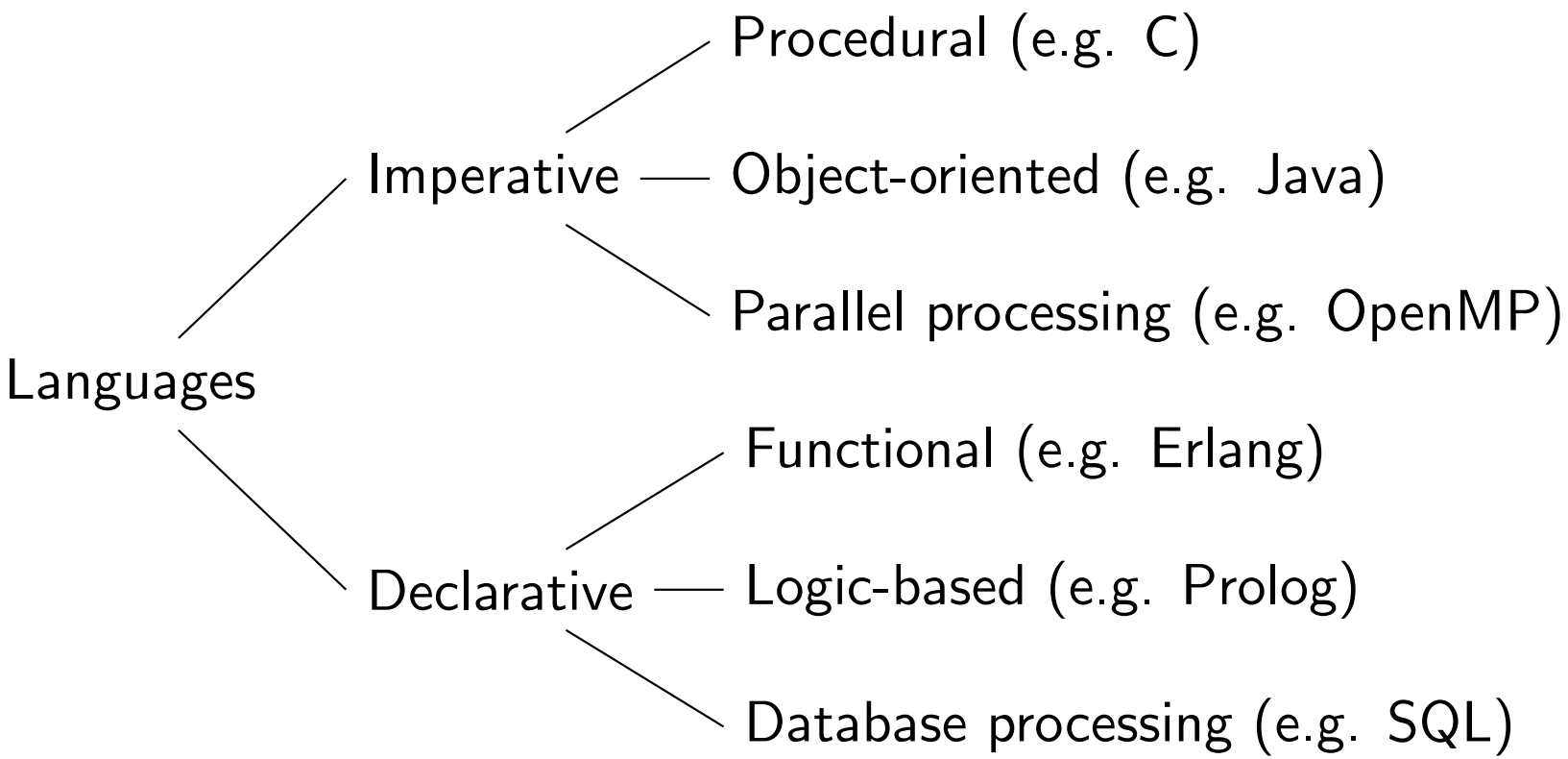
```
function gcd(a, b){
  if b = 0
    return a
  else
    return gcd(b, a % b)
}
```

Recursive (with Modular Division)

```
function gcd(a, b){
  while a != b{
    if a > b
      a <- a - b
    else
      b <- b - a
  }
  return a
}
```

Non-recursive (with Subtraction)

Types of programming languages



Note: Programs specify how it is to be done and what is to be done in the imperative and declarative languages, respectively.

Imperative vs declarative programming language

Imperative	**Declarative**
1. Programs specify how it is to be done. 2. It tells the computer the steps to be taken to obtain a result. 3. It uses statements that change a program's state.	1. Programs specify what is to be done. 1. It tells the computer what result it wants. 2. It expresses the logic of a computation without describing its control flow.

Procedural vs object-oriented programming language

Procedural	**Object-oriented**
1. It takes a top-down approach to divide a program into smaller parts (known as functions).	1. It uses objects to represent everything in a program.
2. It treats data and methods separately.	2. It encapsulates data and methods together.
3. It is less secure.	3. It is more secure.

Functional vs logic programming language

Functional	**Logic**
1. Program evaluation is one-way.	1. Program evaluation is two-way.
2. It uses a virtual machine on which functions operate.	2. It performs applies query on a special domain.
3. It avoids state and mutable data.	3. It extracts knowledge from basic facts and relations.

Compiler vs interpreter

Compiler	**Interpreter**
1. It processes the entire program at a time.	1. It processes a single instruction at a time.
2. It transforms the source code into machine readable instructions.	2. It interprets the source code into executables.
3. It generates intermediate object code, hance memory requirement is more.	3. It does not generate intermediate object code, hence memory requirement is less.
4. It is relatively faster.	4. It is relatively slower.

Homework

- Show flowcharts corresponding to the different versions (non-recursive modular division based, recursive modular division based, non-recursive subtraction based) of the Euclid's algorithm.
- Show a flowchart corresponding to the bisection method.
- Show a flowchart corresponding to the regula falsi method.