Lecture 2

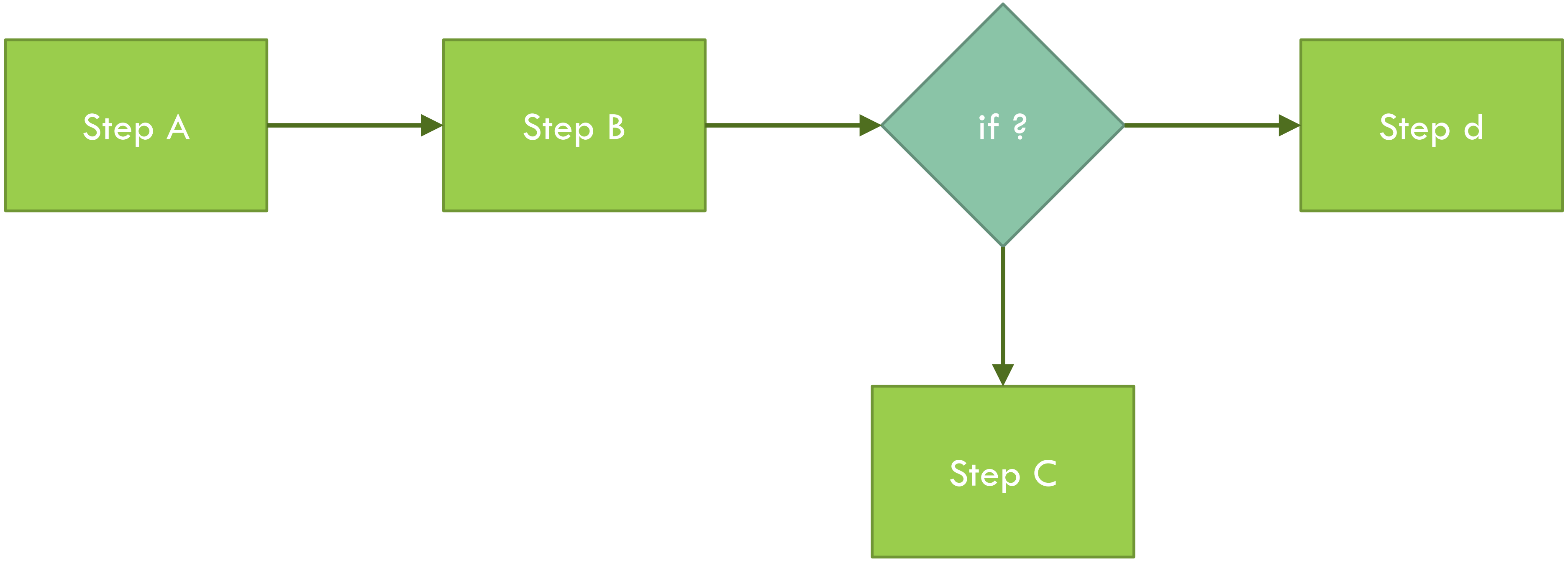
ALGORITHMS AND TESTING

Today’s class:

- Algorithms
- Work on ICE 1 today …hopefully your coding environment is up and running.

Algorithm is a finite set of rules that give a sequence of operations for solving a specific type of problem

- Informally → a recipe, process, method, procedure or routine



Formal Algorithm Description

1. **Finiteness:** an algorithm should terminate after a finite number of steps
   - A procedure that has all the other characteristics of an algorithm except finiteness is a computational method
2. **Definiteness:** each step of an algorithm must be precisely defined
   - Not just "compute the average"
   - Compute the unbiased sample mean, *m*, of a list of *n* integers i_1, i_2,…i_n $m = \frac{1}{n-1}\sum_{l=1}^{n} i_l$
3. **Input:** an algorithm has *zero* or *more* quantities given to it initially or as it runs
4. **Output:** an algorithm has one or more outputs specifically related to the inputs

Input → Algorithm → Output

5. **Effectiveness:** an algorithm's operations should be sufficiently basic to be done exactly in a finite length of time.

Euclids greatest common divisor (GCD) algorithm

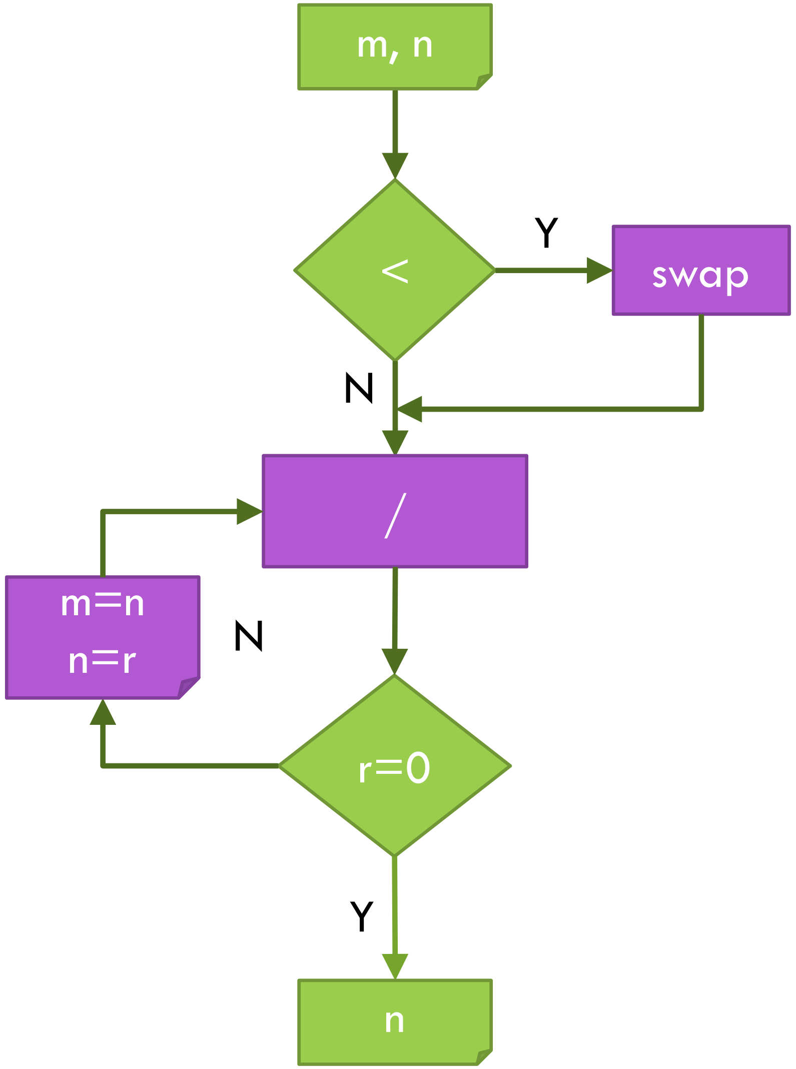
Algorithm *Euclid.* Given two positive integers m and n, find the greatest common divisor, that is, the largest positive integer that evenly divides both m and n

```
A.0 If $m < n$, swap m and n
A.1 Divide m by n and let r be the remainder.
A.2 If $r = 0$, terminate; n is the answer.
A.3 Set m to n, n to r, and go back to step A.1
```

Do the tests!

GCD Algorithm? Do the tests…

- Finiteness:
  - Does the algorithm definitely terminate?
- Definiteness:
  - Did we specify each step?
- Inputs:
  - What are the inputs?
- Outputs:
  - What are the outputs?
- Effectiveness:
  - Are the steps sufficiently basic?



The ADT Bag

- Consider a virtual “bag” as an abstract data type.
  - We are specifying an abstraction inspired by an actual physical bag
  - Basic function to contain its items
  - Can be unordered and possibly contain duplicate objects
  - We insist objects be of same or similar types
- Knowing just its interface
  - Can use ADT bag in a program

```
construct(): construct and empty bag
destroy(): destroy the bag and any contents
add( Item ): add an Item to the bag, returns true on success, else false
remove( Item ): remove a single instance of Item from the bag, returns true on success, else false
isEmpty(): returns true if the bag has no contents, else false
getCurrentSize(): returns the number of items in the bag as an integer
clear(): removes all items in the bag
getFrequencyOf( Item ): the number of times Item appears in the bag
contains( Item ): returns true if at least one Item is in the bag, else false
```

Q: What is the difference between a struct and a class in C++?

A. There is no difference.

B. Structs cannot contain methods, classes can contain methods

C. Default visibility is private for classes and public for structs

D. Structs are only for C not C++

Q: Why would I pass a parameter by constant reference?

A. To make things complicated

B. So that the parameter can be changed by the function

C. To ensure that inadvertent changes to the parameter are caught by the compiler

D. It is always faster

E. To save a copy from being invoked

F. B and E

G. C and D

H. B and D

I. C and E

Assignment/Homework

- Reading : Carrano pp. 37-46
- Complete ICE 1 testing cases, due Tuesday.
- HW1: Chapter 1 exercises 2 and 9, due on Tuesday.
- P1 released and due on Thursday

Practical: Let’s write some more Bag tests and get them to compile and run

- Download the starter code from the website
- Open `bag_tests.cpp` and add some tests for each method of bag.
- Build and run the tests (they should fail)