Programming Languages

Note Title 1/18/2012

Syllabus Overview

First Question:

What programming languages have you used?

- Python
- C++
- Java
- Ruby
- Javascript
- QBasic
- C#
- Matlab
- Php
- C
- Assembly

- VB

Categories

High-level versus low-level:

$$\text{assembly} \xrightarrow{\langle \text{assembler} \rangle} \text{machine code}$$

$$\text{high-level} \xrightarrow{\langle \text{compiler} \rangle} \text{machine or assembly}$$

High-Level Langauges

- Began in 1950's with Fortran
- First machine-independant solution
- Slow to become popular, because compilers were not as good as humans

(Not true now - plus, labor costs more than hardware!)

Why so many?

- Evolution: Still very new!
  - Structured programming (using loops instead of go-tos) was only developed in the late 60's.
  - Object orientation was developed in the '80's.
- Personal preference

- Special purposes: Often, the choice depends on what you want to do!
  - C is good for low level systems work
  - Prolog is good for logical relationships among data
  - Awk is good for character + string manipulation
  - Python + perl are good scripting tools

Other issues

- Learning curve
- Ease of use
- Standardization
- Open Source
- Good Compilers
- Economics + patronage
- Inertia

Families of high-level Languages

① Declarative Languages:
Focus is on what the computer should do

② Imperative Languages:
Focus is on how the computer should do it

★ C++, C, Java, ...

Imperative

Categories:

Ⓐ von Neumann : Fortran, C, Ada.
- based on computation with variables

Ⓑ Scripting languages: bash, awk, php, perl, python, Ruby, etc.
- subset of von Neuman, but tailored for ease of expression over speed

Ⓒ Object-oriented: traced from Simula 67.
often related to von Neuman, but object-based

Declarative

Categories & Examples:

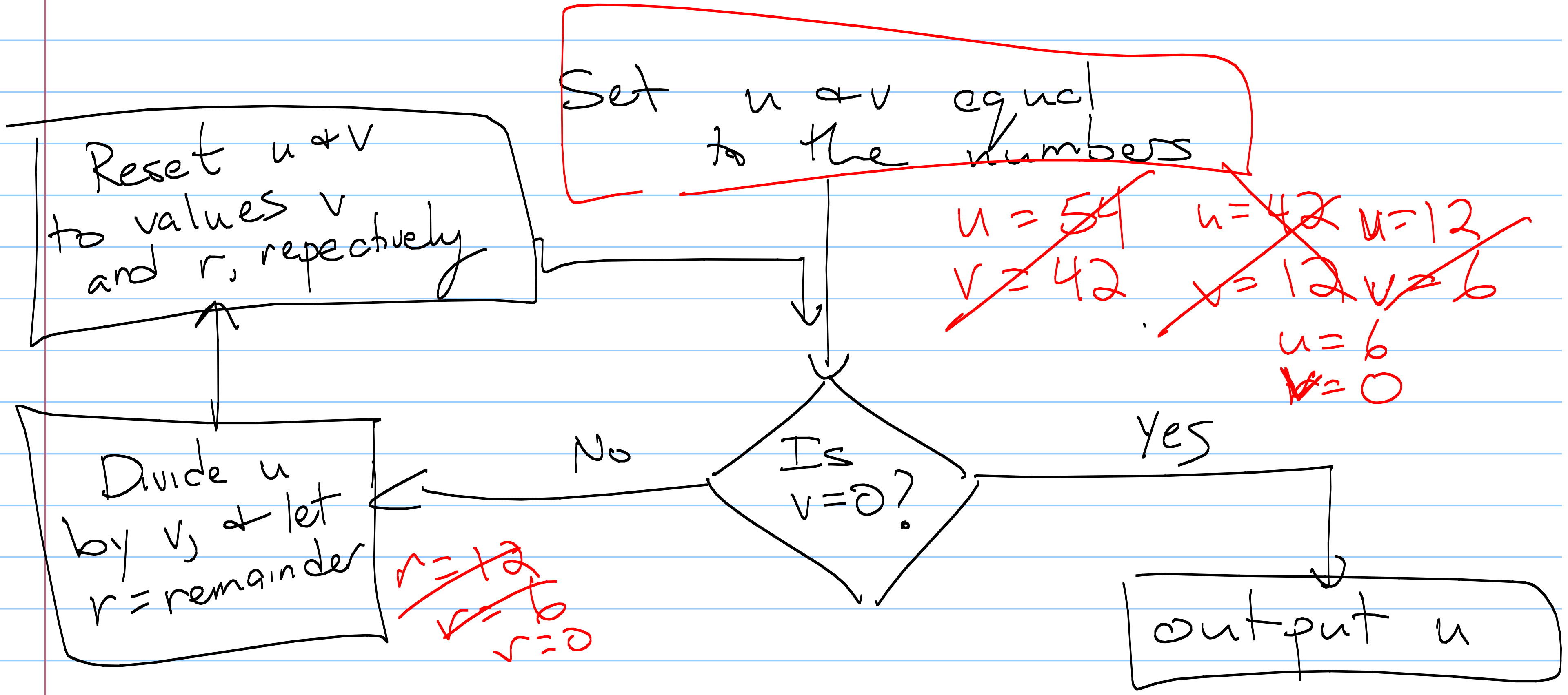
Ⓐ Functional languages: Lisp, Scheme, ML, Haskell
- based on recursive definition of functions (inspired by lambda calculus)

Ⓑ Logic-based: prolog, SQL(?)
- computation is based on attempts to find values that satisfy specified relationships

Ⓒ Data flow: Id, Val
- flow of information (tokens) among nodes

42 & 54
gcd? 6

Example: Compute the gcd
(stolen from my 150 lecture)



$f(n) = f(n-1) + f(n-2)$

GCD in a functional language

$$\gcd(a,b) := \begin{cases} a & \text{if } a=b \\ \gcd(b, a-b) & \text{if } a>b \\ \gcd(a, b-a) & \text{if } b>a \end{cases}$$

$a=54$
$b=42$

$\gcd(54, 42) = \gcd(42, 12)$
$= \gcd(12, 30)$
$= \gcd(12, 18) = \gcd(12, 6)$

GCD in Prolog

gcd(a, b, g) is true if:

- $a = b = g$

there exists

- $a > b$ and $\exists c$ such that $c = a - b$ and gcd(c, b, g) is true

- $b > a$ and $\exists$ c s.t. $c = b - a$ and gcd(c, a, g) is true