Telling a computer how to behave
(via pseudocode -- a workaround for Computing's Tower of Babel.)

COS 116: 2/13/2007

Adam Finkelstein

The Economist

Face value

St Lawrence of Google

Jan 12th 2006

Paul Saffo at Silicon Valley's Institute for the Future says that “Google is a religion posing as a company.”

Playing God

If Google is a religion, what is its God?

It would have to be The Algorithm.

Recall: Scribbler

Inputs

Stall sensor

Line sensor (underneath)

Light sensors

Obstacle sensor detector

Obstacle sensor emitter

Outputs

Speaker

Motor/wheels

Light outputs

Recall: Scribbler's "Language"

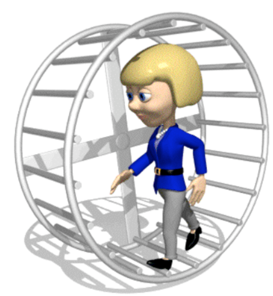
- Several types of simple instructions
  - E.g. "Move forward for 1 s"
- Two types of compound instructions

Conditional (a.k.a. Branching)

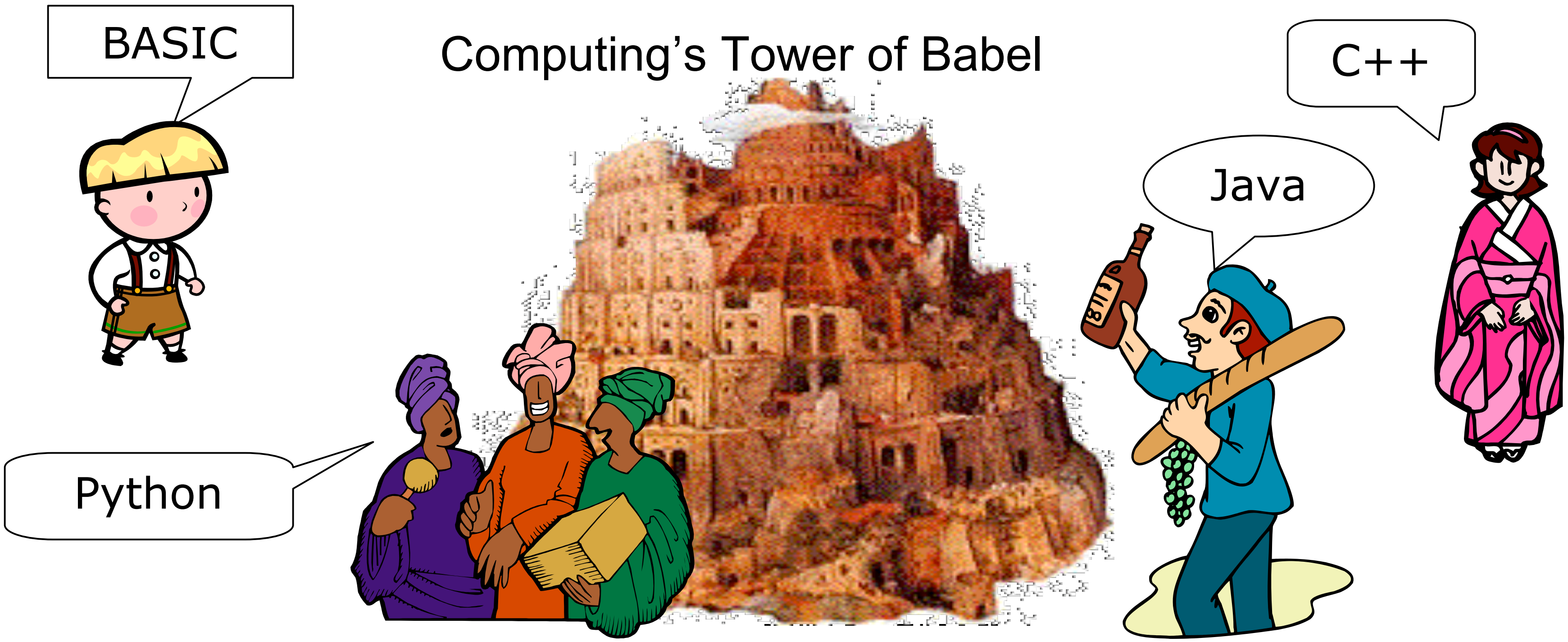
If <condition> Then
{
List of instructions
}
Else
{
List of instructions
}

Loop

Do 5 times
{
List of instructions
}



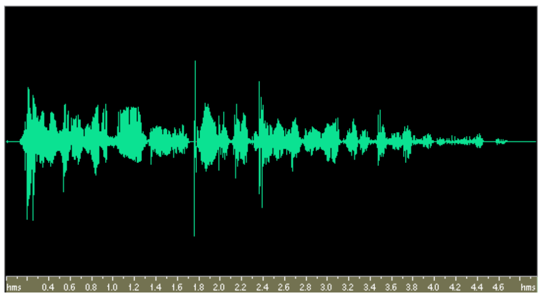
Scribbler language illustrates essential features of all computer languages



- Fundamental features of human languages:nouns/verbs/adjectives, subjects/objects, pronouns, etc.
- Computer languages also share fundamental features, e.g. conditional and loop statements, variables, **ability to perform arithmetic**, etc.

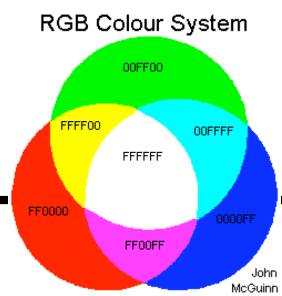
For a computer, everything's a number

Audio waveform



Sequence of Numbers representing frequency, amplitude, etc.

Image



Sequence of Numbers representing red/green/blue color value of each pixel.

A simple problem

- Our robot is getting ready for a big date…

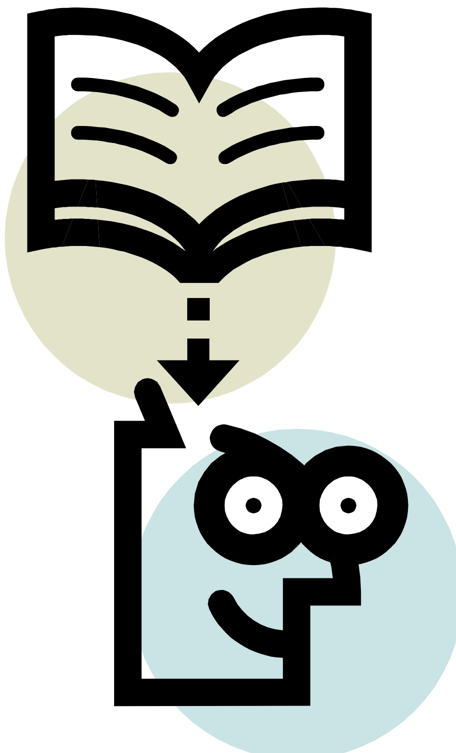
- How would it identify the cheapest bottle? (Say it can scan prices)

Solution

- Pick up first bottle, check price
- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.

Similar question in different setting

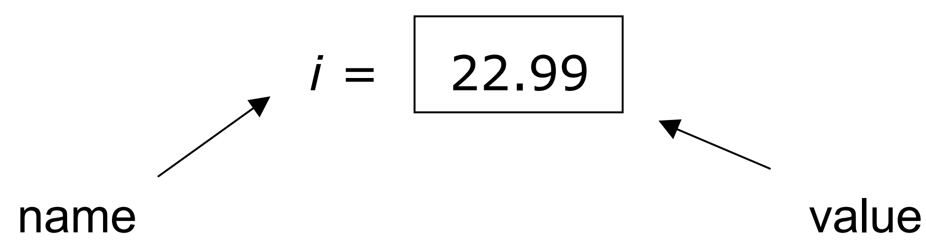
- Robot has *n* prices stored in memory
- Want to find minimum price



Memory: a simplified view

- A scratchpad that can be perfectly erased and re-written any number of times

- A variable: a piece of memory with a name; stores a “value”



Examples

$i \leftarrow 5$	Sets *i* to value 5
$j \leftarrow i$	Sets *j* to whatever value is in *i*. Leaves *i* unchanged
$i \leftarrow j + 1$	Sets *i* to $j + 1$. Leaves *j* unchanged
$i \leftarrow i + 1$	Sets *i* to 1 more than it was.

Arrays

- *A* is an array of *n* values, *A*[*i*] is the *i*'th value

A =	40.99	62.99	52.99	...	22.99

- Example: *A*[3] = 52.99

Solution

- Pick up first bottle, check price

- Walk down aisle. For each bottle, do this:
  - If price on bottle is less than price in hand, exchange for one in hand.

Procedure findmin

- n items, stored in array A
- Variables are i, $best$
- $best \leftarrow 1$
- Do for i = 2 to n

  {

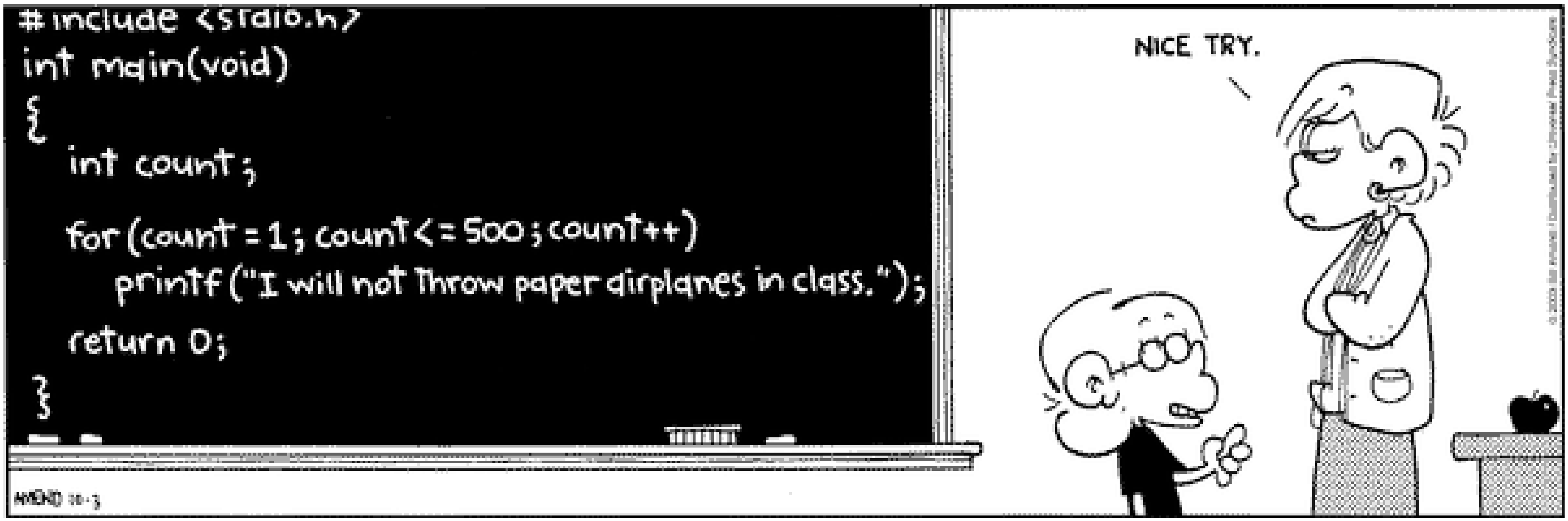
  if ($A[\,i\,] < A[best]$) then

  { $best \leftarrow i$ }

  }

Another way to do the same

```
best ← 1;
i ← 1
Do while (i < n)
{
        i ← i + 1;
        if ( A[ i ] < A[best] ) then
                { best ← i }
}
```


#include <stdio.h>
int main(void)
{
int count;
for(count = 1; count<=500; count++)
printf("I will not throw paper airplanes in class.");
return 0;
}
NICE TRY.
AMEND 10-3

New problem for robot: sorting

Arrange them so prices <u>increase</u> from left to right.

Solution

Do for i=1 to n-1

{

Find cheapest bottle among those numbered i to n

Swap that bottle and the i'th bottle.

}

"selection sort"

Swapping

- Suppose x and y are variables.
  How do you <u>swap</u> their values?

- Need extra variable!

$$tmp \leftarrow x$$
$$x \leftarrow y$$
$$y \leftarrow tmp$$

Algorithm

- A precise unambiguous procedure for accomplishing a task

- Named for Abu Abdullah Muhammad bin Musa al-Khwarizmi
  - His book "Al-Jabr wa-al-Muqabilah" evolved into today's high school algebra text.

- Examples:
  recipe, long division, selection sort.

Love, Marriage, and Lying

Standard disclaimer.

Stable Matching Problem

- Problem: Given N men and N women, find a "suitable" matching between men and women.
  - Participants rate members of opposite sex.
  - Everyone lists preferences from best to worst.

Men's Preference List

Man	1st	2nd	3rd	4th	5th
Victor	Bertha	Amy	Diane	Erika	Clare
Wyatt	Diane	Bertha	Amy	Clare	Erika
Xavier	Bertha	Erika	Clare	Diane	Amy
Yancey	Amy	Diane	Clare	Bertha	Erika
Zeus	Bertha	Diane	Amy	Erika	Clare

best ↑ (1st) … worst ↑ (5th)

Stable Matching Problem

- Problem: Given N men and N women, find a "suitable" matching between men and women.
  - Participants rate members of opposite sex.
  - Everyone lists preferences from best to worst.

Women's Preference List

Woman	1st	2nd	3rd	4th	5th
Amy	Zeus	Victor	Wyatt	Yancey	Xavier
Bertha	Xavier	Wyatt	Yancey	Victor	Zeus
Clare	Wyatt	Xavier	Yancey	Zeus	Victor
Diane	Victor	Zeus	Yancey	Xavier	Wyatt
Erika	Yancey	Wyatt	Zeus	Xavier	Victor

best worst

Stable Matching Problem

- Problem: Given N men and N women, find a "suitable" matching between men and women.
  - PERFECT matching: everyone matched monogamously.
    - each man gets exactly one woman, and vice-versa
  - STABILITY: no incentive for some pair of participants to undermine assignment by joint action.
    - a pair that is <u>not</u> matched with each other is UNSTABLE if they prefer each other to current partners
    - unstable pair: each improve by dumping spouses and eloping
- STABLE MATCHING (Gale and Shapley, 1962) = perfect matching with no unstable pairs.

Example

Men's Preference List

Man	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

Women's Preference List

Woman	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

- Lavender assignment is a perfect matching. Are there any unstable pairs?

Yes. Bertha and Xavier form an unstable pair. They would prefer each other to current partners.

Example

Men's Preference List

Man	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

Women's Preference List

Woman	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

- Green assignment is a stable matching.

Example

Men's Preference List

Man	1st	2nd	3rd
Xavier	A	B	C
Yancey	B	A	C
Zeus	A	B	C

Women's Preference List

Woman	1st	2nd	3rd
Amy	Y	X	Z
Bertha	X	Y	Z
Clare	X	Y	Z

- Gray assignment is also a stable matching.

Propose-And-Reject Algorithm

- Guarantees a stable matching.

Gale-Shapley Algorithm (men propose)

```
Initialize each person to be free.
while (some man m is free and hasn't proposed to every woman)
    w = first woman on m's list to whom he has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé f)
        assign m and w to be engaged, and f to be free
    else
        w rejects m
```

Understanding the Solution

- For a given problem instance, there may be several stable matchings.
  - Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

- Yes. Gale-Shapley finds MAN-OPTIMAL stable matching!
- Gale-Shapley finds WOMAN-PESSIMAL stable matching.

Extensions

- Unacceptable partners
  - Every woman is not willing to marry every man, and vice versa.
  - Some participants declare others as “unacceptable.”
- Sets of unequal size
  - Unequal numbers of men and women, e.g. 100 men & 90 women
- Limited Polygamy
  - e.g., Bill wants to be matched with 3 women.

Matching Residents to Hospitals

- Hospitals ~ Men (limited polygamy allowed).
- Residents ~ Women (more than hospitals)
- Started just after WWII (before computer usage).
- Ides of March, 13,000+ residents are matched.
- Rural hospital dilemma.
  - Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
  - How to find stable matching that benefits rural hospitals?

Lessons Learned

- Powerful ideas learned in computer science.
- Sometimes deep social ramifications.
  - Hospitals and residents…
  - Historically, men propose to women.
    Why not vice versa?
  - Men: propose early and often.
  - Women: ask out the guys.
  - Computer scientists get the best partners!!!