CSC 171
Introduction to
Computer Programming

Dr. Stephen Bloch
sbloch@adelphi.edu
http://www.adelphi.edu/sbloch/class/171/

What's this course about?

- Fundamentals of computer programming
- Going from a word problem to a working program
- Designing programs to be modified and reused
- How computer scientists think

What's this course *not* about?

- A particular programming language
  - (e.g. C++, Java, Scheme, perl, ...)
- A particular programming environment
  - (e.g. VisualStudio, DrScheme, ...)
- Designing Web pages
- Arithmetic

However, we have to use...

- A programming language
  - (we'll use Scheme, and later Java)
- A programming environment
  - (we'll use DrScheme, and later ProfessorJ)
- The Web
  - (for textbook, assignments, announcements, ...)
- Arithmetic
  - (as an example you've all seen)

Who should take this course?

- CS and CMIS majors
- Math majors for programming credit
  (can take 160 instead)
- Other majors curious about computer programming
  (can take 160 instead)

Who should *not* take this course?

- End-users and Web page authors
- Students looking for a “cake” course

3 hours/week lecture
3 hours/week lab
6 hours/week homework & reading

Imagine this assignment...

- 20-page paper
- Due at end of semester
- On Napoleon's invasion of Russia

What kinds of knowledge are needed?

- Napoleon & Russia
- How to write a 20-page paper
- How to finish a long-term project on time

Imagine this assignment...

- 20-page paper
- Due at end of semester
- On Napoleon's invasion of Russia
- In Swedish
- With a quill pen

What kinds of knowledge are needed?

- Napoleon & Russia
- How to write a 20-page paper
- How to finish a long-term project on time
- Swedish language (spelling, vocabulary, grammar, idioms…)
- How to use a quill pen

In a programming course...

- Content-area knowledge
  - (graphics, economics, physics, etc.)
- How to structure a program
- How to finish a long-term project on time
- Scheme (or C++ or Java or whatever) language
- How to use the software & hardware

What's really going on inside ordinary arithmetic?

$$3 + 4 \longrightarrow 7$$

What's really going on inside ordinary arithmetic?

$$\boxed{\boxed{3} \oplus \boxed{4}} \longrightarrow 7$$

What's really going on inside ordinary arithmetic?

$\boxed{\boxed{3} \oplus \boxed{4}} \longrightarrow 7$

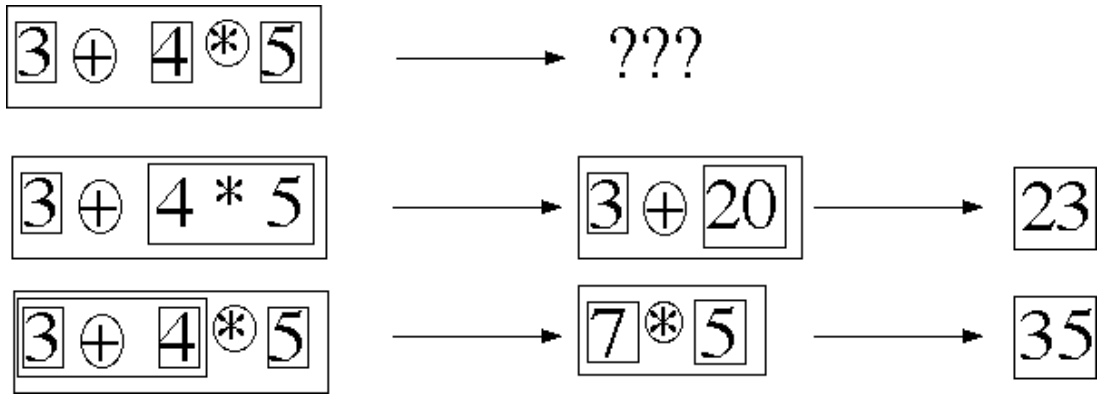
3 + (4 * 5)

What's really going on inside ordinary arithmetic?

$$\boxed{\boxed{3} \oplus \boxed{4}} \longrightarrow 7$$

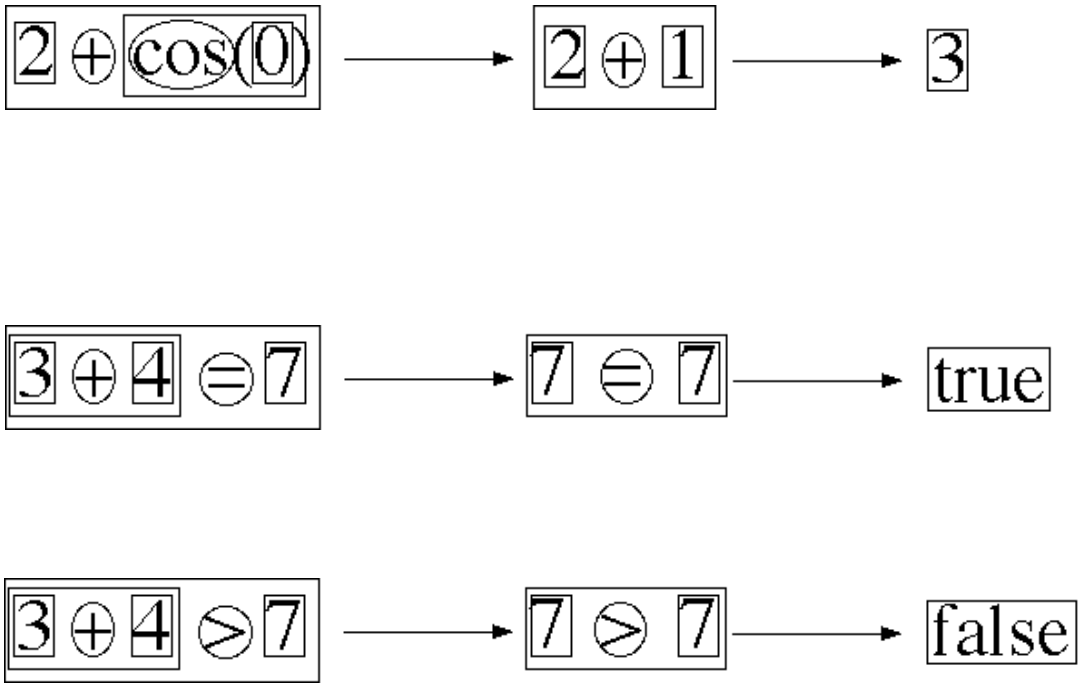
$$\boxed{\boxed{3} \oplus \boxed{(\boxed{4} \circledast \boxed{5})}} \longrightarrow \boxed{\boxed{3} \oplus \boxed{20}} \longrightarrow \boxed{23}$$

Ambiguity



We usually resolve the ambiguity with
precedence rules:
PEMDAS

More arithmetic



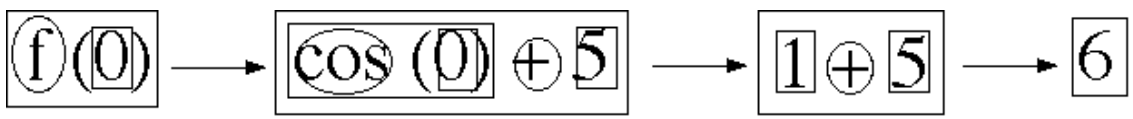
Definitions in arithmetic

Let x = $\boxed{7}$ $\longrightarrow$ has no value, but a side effect instead

$$\boxed{\boxed{6}\ \boxed{x} \oplus \boxed{2}} \longrightarrow \boxed{\boxed{6} \circledast \boxed{7} \oplus \boxed{2}} \longrightarrow \boxed{\boxed{42} \oplus \boxed{2}} \longrightarrow \boxed{44}$$

Function definitions in arithmetic

- Let f(x) = cos(x) + 5
  - (has no value, but a side effect instead)



A lot of this is confusing

1) Some operators go *between two* operands; others go *before one* operand; some require parentheses around operand, some don't.

2) Need precedence rules (PEMDAS) to resolve ambiguity

3) Sometimes = means "are these equal?"; sometimes it means "define this to stand for that".

4) Sometimes there's no visible operator; defaults to multiplication

5) (3+4) means the same as 3+4, or even ((3+4))

Some simpler, more consistent rules:

1) All operators go *before* however many operands they need.

2) All subexpressions *must* have parentheses around them (including the operator).

3) = means “are these equal?”;
define means “make this stand for that”.
Both count as operators (see rules 1 & 2)

4) No hidden operators; if you mean *, say it.

5) No extra parentheses allowed; *exactly one* pair of parentheses per operator.

Old vs. new notation

- 3 + 4
- 3 + 4 * 5
- (3+4) * 5
- 2 + cos(0)
- 3 + 4 = 7
- 3 + 4 > 7
- Let x = 7
- 6x + ((2))
- Let f(x) = cos(x) + 2
- f(0)

- (+ 3 4)
- (+ 3 (* 4 5))
- (* (+ 3 4) 5)
- (+ 2 (cos 0))
- (= (+ 3 4) 7)
- (> (+ 3 4) 7)
- (define x 7)
- (+ (* 6 x) 2)
- (define (f x) (+ (cos x) 2))
- (f 0)