Programming Languages: Syntax and Semantics

1. Why not program in a Natural Language like English?
2. Why are there so many programming languages
3. Syntax vs Semantics
4. Languages and Hardware
5. Compilers, Assemblers and Interpreters

1. Why not program in a Natural Language like English?

Natural languages have evolved to be rich, nuanced, and... ambiguous

A: How do you make a turtle fast?

1. Why not program in a Natural Language like English?

Natural languages have evolved to be rich, nuanced, and... ambiguous

A: How do you make a turtle fast?

B: Take away its food.

1. Why not program in a Natural Language like English?

Natural languages have evolved to be rich, nuanced, and... ambiguous

A: I saw a man eating shark at the aquarium!

1. Why not program in a Natural Language like English?

Natural languages have evolved to be rich, nuanced, and... ambiguous

A: I saw a man eating shark at the aquarium!

B: That's nothing. I saw a man eating herring at the deli!

1. Why not program in a Natural Language like English?

Natural languages have evolved to be rich, nuanced, and... ambiguous

A: Time flies like an arrow.

1. Why not program in a Natural Language like English?

I. Natural languages have evolved to be rich, nuanced, and... ambiguous

A: Time flies like an arrow.

B: Yeah, and fruit flies like a banana!

We don't want programming languages to be ambiguous (mostly).

- Our software has REQUIREMENTS
- It's no good to say, "Well, the computer thought that the code meant one thing, and the programmers thought something different."

II. PLs are much smaller than NLs, with strict enforcement of grammar rules — this makes compilers + interpreters easier to code.

g++ = 15 million lines of code

it is one of the largest free software projects in existence

= 10-20 MB for the executable

2. Why are there so many programming languages?

(See rosettacode.org/wiki/Category:Programming_Languages)

2. Why are there so many programming languages?

(See rosettacode.org/wiki/Category:Programming_Languages)

An aside: Go to this site and put "Hello world" in the search bar... You can see how to write this program in 947 different languages!

2. Why are there so many programming languages?

(See rosettacode.org/wiki/Category:Programming_Languages)

I. Some languages are designed to be tuned to particular purposes:
web development, network coding, Sys Admin,

II. Some languages are designed to enhance certain functionality eg threading, ease-of-learning, robust error-checking, etc

III Some languages offer different ways of thinking! (ie. ways of thinking about problem solving)
eg Imperative (C++), logic (Prolog), list-based/recursive (Lisp)

3. Syntax vs Semantics

Syntax = the "grammar" of what needs to be written to invoke the desired computation

Semantics = the computation that results

SYNTAX

SEMANTICS

Eg: =1/2 * 10

that is syntactically correct....

... in Excel, as a cell's content 5

... in C++, as the RHS of an assignment 5

4. Languages and Hardware

One can think of the computer's contents as being one long string of 0's and 1's ....

0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 0

The computer accesses the "bits" in segments, called words Each word has an address and is of a fixed size (which is a part of the processor's design).

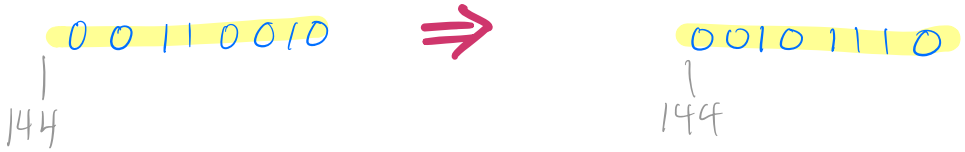
When the computer is told to run a.out it goes to the address for that program and reads the "words" as instructions. a.out = 103

0 1 1 0 1 0 0 1 — on this processor, 0110 1001 means MOVE

1 0 0 1 0 0 0 0 - the Move is always followed by a memory location - a "variable"

0 0 1 0 1 1 1 0 - the new value to be "moved" into this memory location is 46

Effect in memory:



Machine language (different for each type of processor)

10110101 11010000 00001111 .....

Assembly Language

MOV Mem208 #15

People do write code in Assembler!
But we have also invented HLLs
(High Level Languages) where a single
HLL instruction may engender hundreds
or thousands of lines of assembler.

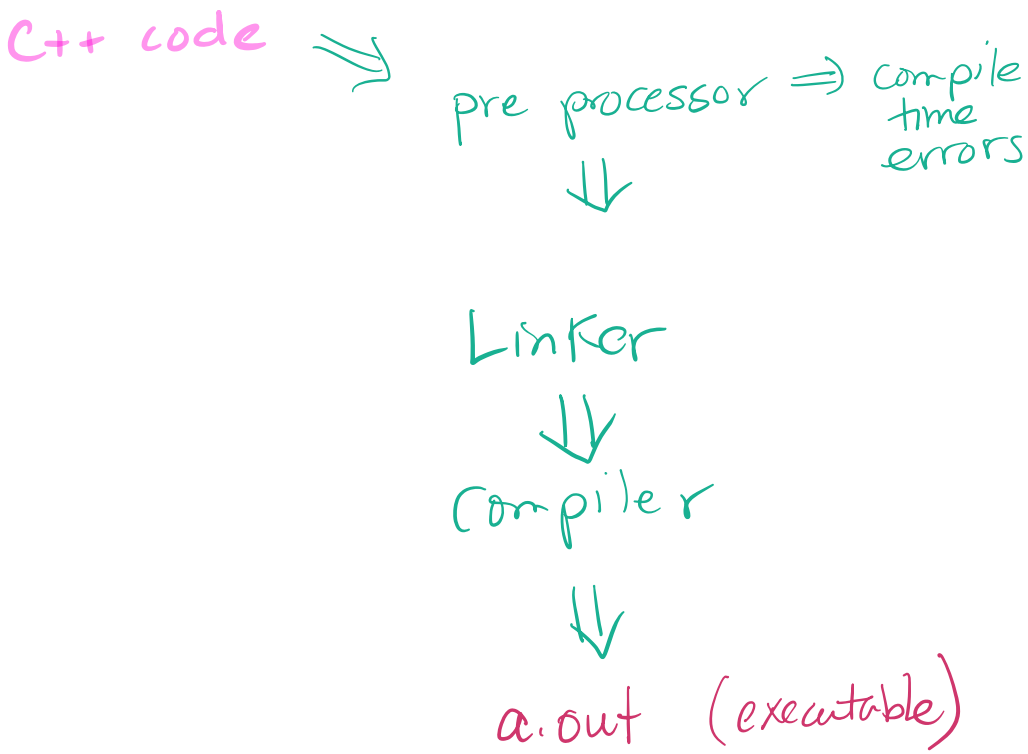
An HLL like C++ is "in between"

machine language ← C++ → Natural Language

It's pretty easy for a computer to understand
(after a compiler translates it), and
it's pretty easy for a human to understand
(with training... like CSCI 159)

A compiler takes a C++ program and translates it into Machine Code

An interpreter does the same, but it does so "in real time" while the user interacts with it.


C++ code
pre processor
compile time errors
Linker
Compiler
a.out (executable)