

# A gentle introduction to Programming Languages

Evan Misshula

February 7, 2017

## Contents

### Programming languages help us think



## History

First computers had no languages except machine code

- Euclid's algorithm for GCD in machine code

## Machine code

```
55 89 e5 53 83 ec 04 83 e4 f0 e8 31 00 00 00 89 c3 e8 2a 00
00 00 39 c3 74 10 8d b6 00 00 00 00 39 c3 7e 13 29 c3 39 c3
75 f6 89 1c 24 e8 6e 00 00 00 8b 5d fc c9 c3 29 d8 eb eb 90
```

## Assembler code

```

    pushl    %ebp
    movl     %esp, %ebp
    pushl    %ebx
    subl     $4, %esp
    andl     $-16, %esp
    call     getint
    movl     %eax, %ebx
    call     getint
    cmpl     %eax, %ebx
    je       C
A:    cmpl     %eax, %ebx
                                jle     D
                                subl     %eax, %ebx
                                B:    cmpl     %eax, %ebx
                                jne     A
                                C:    movl     %ebx, (%esp)
                                call     putint
                                movl     -4(%ebp), %ebx
                                leave
                                ret
                                D:    subl     %ebx, %eax
                                jmp      B
```

## Limitations

- Each language depended on the machine
- So detail oriented that it was hard to keep track of

## Desire for machine independence

- Invention of Fortran
- Literally "Formula Translation"
- First compiled language

## Definition of a compiler

- A compiler is a piece of software that translates a higher level language into machine code.

## Limitations of compiled languages

- Humans could always write programs that beat the compiler

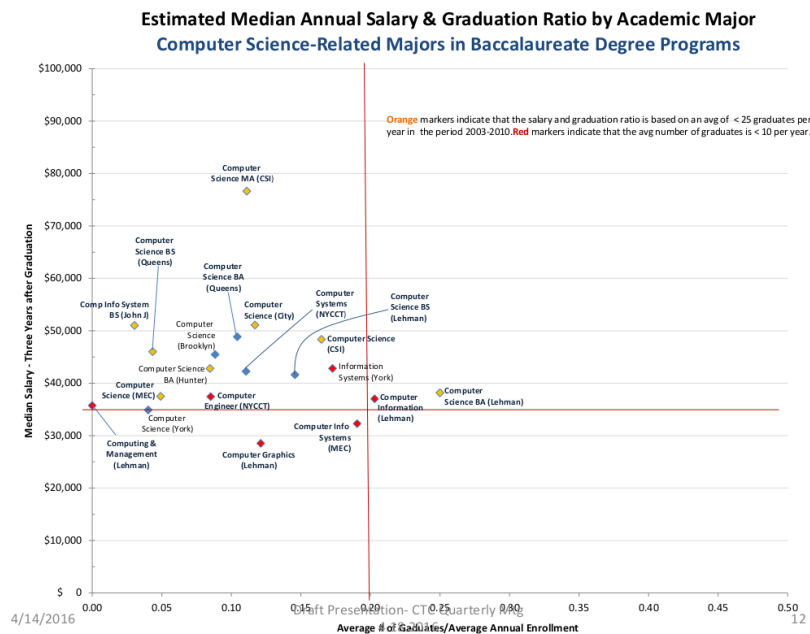
## Now compilers always beat humans

- Technology improved and compiled code is either better or no worse than human optimized code.

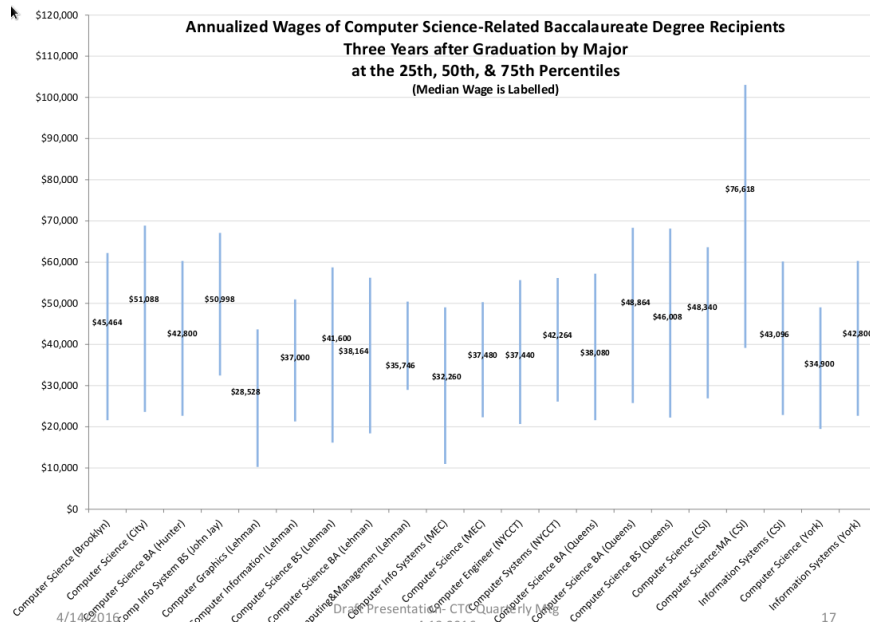
## Why is all of this important to you

- BLS Statistics
- 2015 median salary is \$100,690
- Number of jobs: 1,114,000
- Job growth: 17% (much faster than average)

## Salaries vs Graduation



## Salaries Distribution



## Languages developers and employers want

- The most important job is to know

<http://stackoverflow.com/research/developer-survey-2016#technology-most-loved-dreaded->

## Program lifecycle phase

- Edit time
- Compile time
- Distribution time
- Installation time
- Link time
- Load time
- Run time

## **Edit time**

- This is while we are editing the program. This is the phase that version control allows us to collaborate in.

## **Compile time**

- This is the phase when our program is being translated to machine code. This is when a type system may throw errors.

## **Distribution time**

- The time that it takes to distribute a program. Think of downloading from a git repo, CRAN, Ubuntu or Debian repository.

## **Installation time**

- The time it takes to install the program on the user's system.

## **Link time**

- The time it takes to connect together any dependencies.

## **Load time**

- The time that it takes for a stored image to be placed in memory

## **Run time**

- The time the program is executing.

## **Programming paradigms**

- Imperative
- Procedural
- Object Oriented

- Functional

## Imperative

In computer science, imperative programming is a programming paradigm that uses statements that change a program's state. In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform. Imperative programming focuses on describing how a program operates.

## Procedural

Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call. Procedures, also known as routines, subroutines, or functions (not to be confused with mathematical functions), simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

## Object Oriented

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.

## Functional Programming

In computer science, functional programming treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. It is a declarative programming paradigm, which means programming is done with expressions or declarations instead of statements.

## Functional Programming II

In functional code, the output value of a function depends only on the arguments that are input to the function, so calling a function  $f$  twice with the same value for an argument  $x$  will produce the same result  $f(x)$  each time.