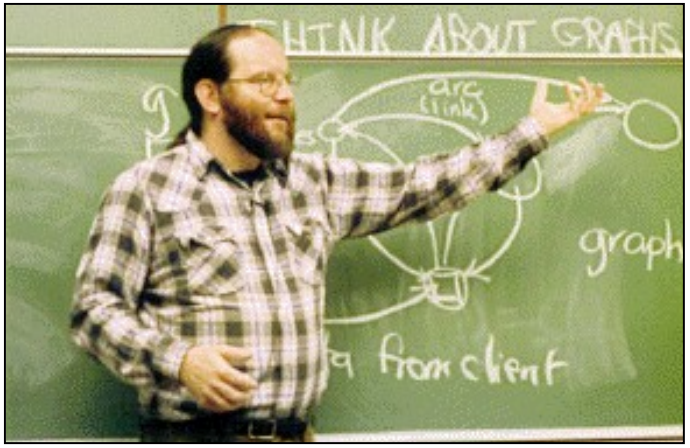
Introduction to CS 106B

Eric Roberts
CS 106B
January 5, 2015

CS 106B Staff



Professor: Eric Roberts
eroberts@cs.stanford.edu
Office Hours (Gates 202):
Tuesdays 9:30-11:30
Wednesdays after class at Bytes

Head TA: Kevin Miller
kmiller4@stanford.edu
Office Hours (Gates 160):
Wednesdays 12:30-2:00
Thursdays 1:00-3:00

Essential Information

CS 106B: Programming Abstractions (ENGR 70B)

Abstraction and its relation to programming. Software engineering principles of data abstraction and modularity. Object-oriented programming, fundamental data structures (such as stacks, queues, sets) and data-directed design. Recursion and recursive data structures (linked lists, trees, graphs). Introduction to time and space complexity analysis. Uses the programming language C++ covering its basic facilities.

Prerequisite: 106A or equivalent.

Terms: Aut, Win, Spr, Sum | Units: 3-5 | UG Reqs: GER:DBEngrAppSci | Grading: Letter or S/NC

Instructors: Lee, C. (PI); Roberts, E. (PI), Stepp, M. (PI)

Schedule for CS 106B

2014-2015 Winter

CS 106B | 3-5 units | UG Reqs: GER:DBEngrAppSci, WAY-FR | Class # 3489 | Section 01 | Grading: Letter or S/NC| LEC

01/05/2015 - 03/13/2015 Mon, Wed, Fri 2:15 PM - 3:05 PM at Hewlett Teaching Center 200 with Roberts, E. (PI)

Instructors: Roberts, E. (PI)

Notes: Same as Eng 70B. May be taken for 3 units by graduate students.

News Flash: CS 106B lectures will be recorded this quarter!

Is CS 106B the Right Course?

CS 106A: Programming Methodology

Introduction to the engineering of computer applications emphasizing modern software engineering principles: object-oriented design, decomposition, encapsulation, abstraction, and testing. Uses the Java programming language. Emphasis is on good programming style and the built-in facilities of the Java language. No prior programming experience required.

Terms: Aut, Win, Spr, Sum | Units: 3-5 | UG Reqs: GER:DBEngrAppSci, WAY-FR | Grading: Letter or S/NC

Instructors: Sahami, M. (PI) ; Schwarz, K. (PI) ; Stepp, M. (PI)

CS 106X: Programming Abstractions (Accelerated)

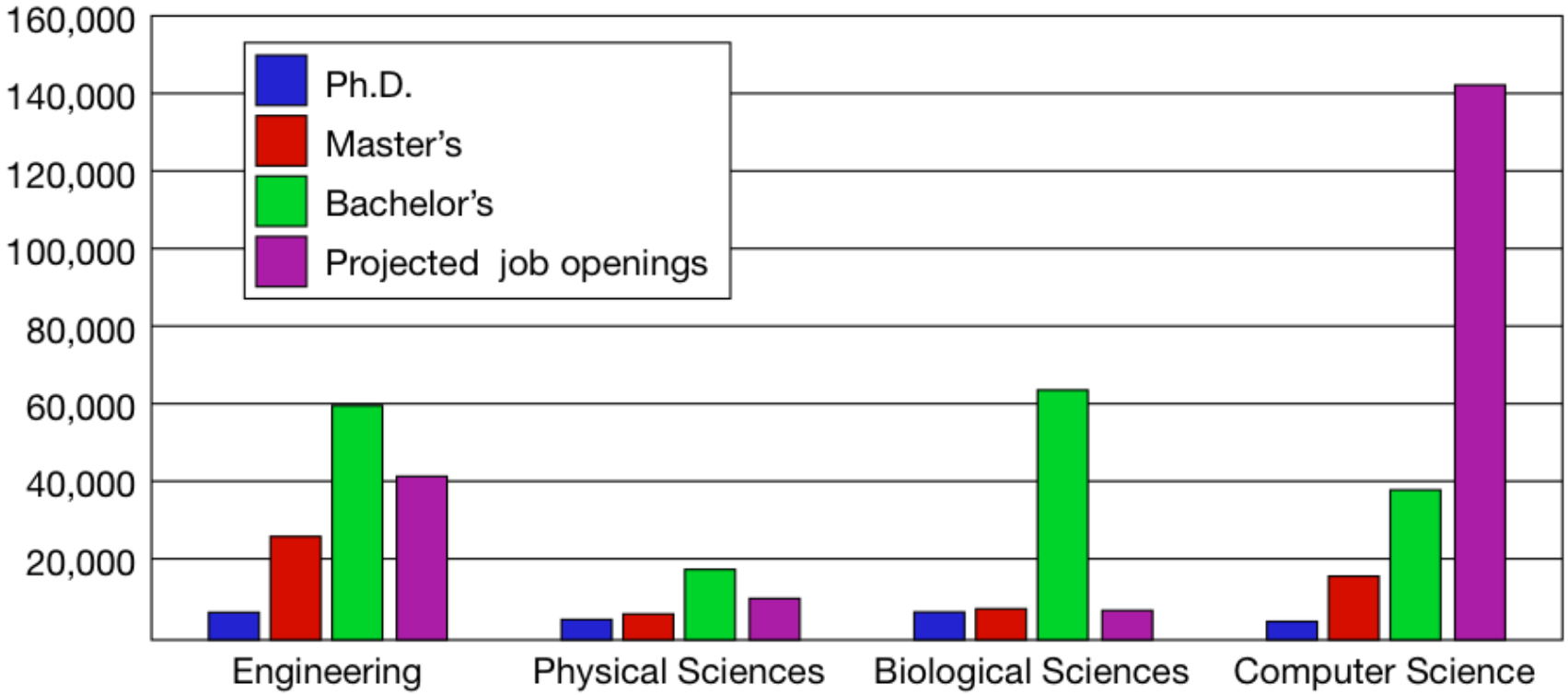
Intensive version of 106B for students with a strong programming background interested in a rigorous treatment of the topics at an accelerated pace. Additional advanced material and more challenging projects.

Prerequisite: excellence in 106A or equivalent, or consent of instructor.

Terms: Aut | Units: 3-5 | UG Reqs: GER:DBEngrAppSci, WAY-FR | Grading: Letter or S/NC

Instructors: Lee, C. (PI)

Degree Production *vs.* Job Openings

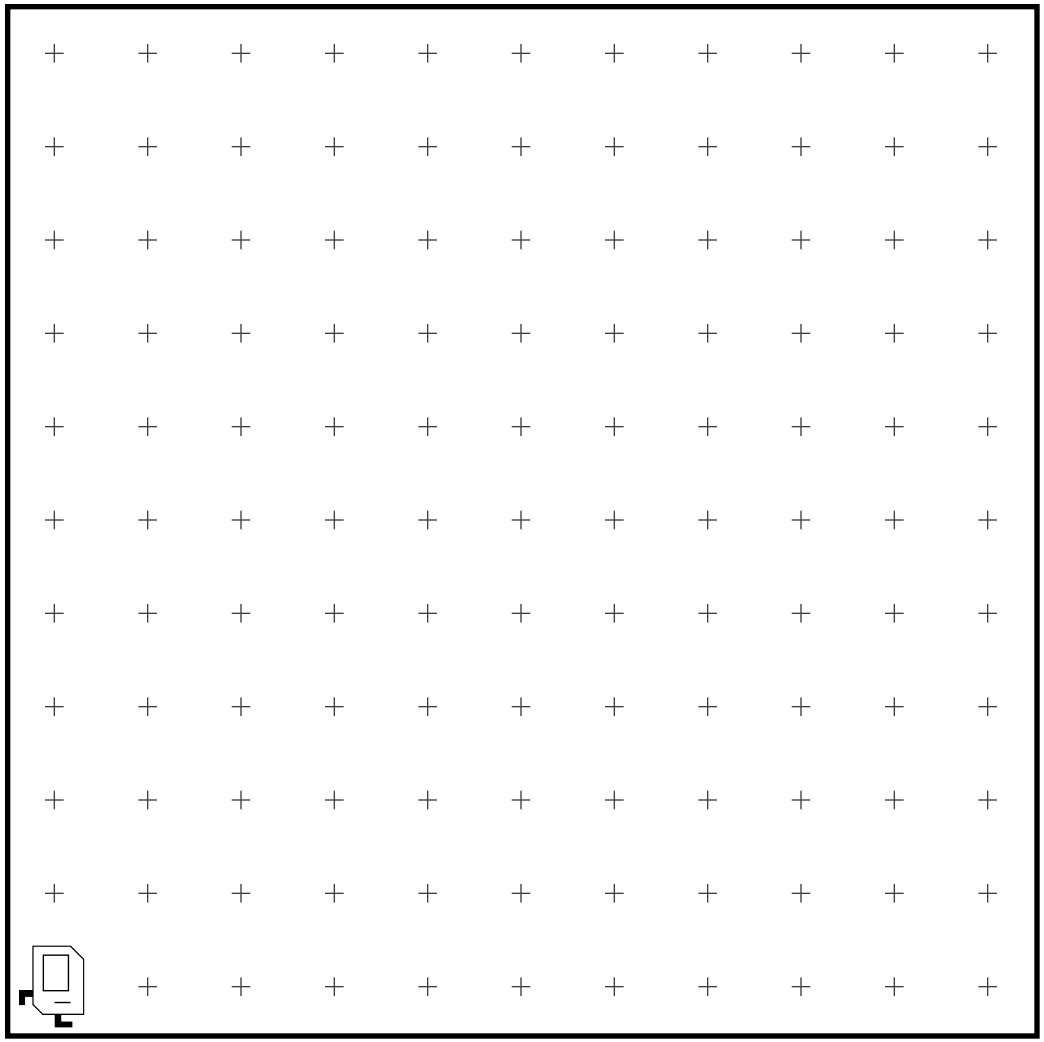


Sources: Adapted from a presentation by John Sargent, Senior Policy Analyst, Department of Commerce, at the CRA Computing Research Summit, February 23, 2004. Original sources listed as National Science Foundation/Division of Science Resources Statistics; degree data from Department of Education/National Center for Education Statistics: Integrated Postsecondary Education Data System Completions Survey; and NSF/SRS; Survey of Earned Doctorates; and Projected Annual Average Job Openings derived from Department of Commerce (Office of Technology Policy) analysis of Bureau of Labor Statistics 2002-2012 projections. See **http://www.cra.org/govaffairs/content.php?cid=22**.

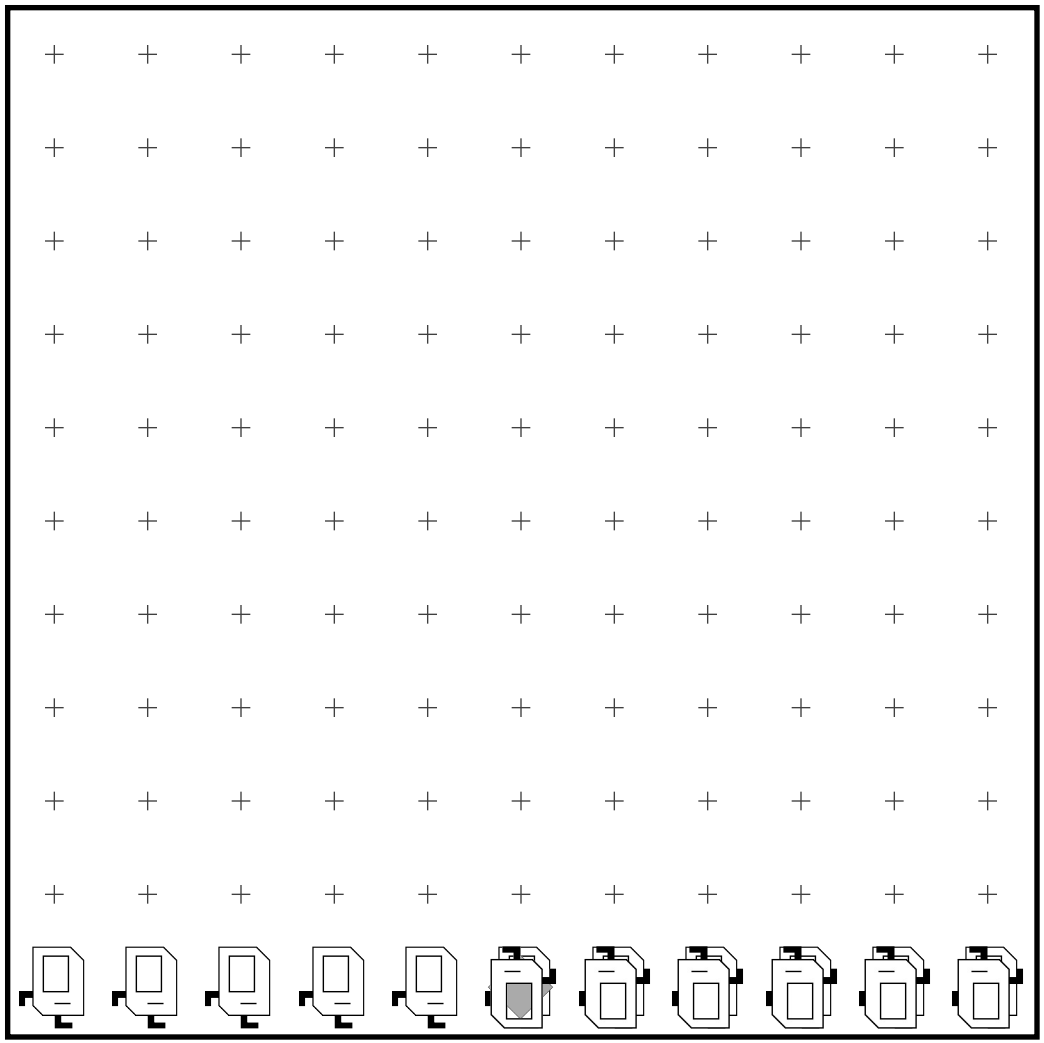
The Big Ideas in CS 106B

- ***Recursion***. Recursion is an enormously powerful technique that enables you to solve complex problems that you would never be able to solve without it.
- ***Data abstraction***. For most of the quarter, we'll be learning about a variety of abstract data types that will prove to be enormously valuable as you write programs.
- ***Algorithmic efficiency***. As you will learn over the course of the quarter, different algorithms can vary enormously in terms of the amount of time required to solve a particular problem. In CS 106B, you will learn how to measure algorithmic efficiency along with some general techniques for developing efficient algorithms.

Find the Midpoint



The Power of Recursion



Syllabus—Week 1

January 5	7	9
Course overview The big ideas in CS 106B The C++ language C++ *vs.* Java	Functions in C++ Call by reference Libraries and interfaces Recursive functions	Using the **`string`** class File streams Class hierarchies
Read: Chapter 1	**Read: Chapters 2 and 7**	**Read: Chapters 3 and 4**

Syllabus—Week 2

12 Abstract data types Using **`Vector`** and **`Grid`** Stacks and queues **Read: Sections 5.1-5.3**	16 **`Map`**, **`Set`**, and **`Lexicon`** Iterating over a collection **Read: Sections 5.4-5.6**	18 Designing classes The **`TokenScanner`** class **Read: Chapter 6** **Due: HW #1 (Simple C++)**

Syllabus—Week 3

19	21	23
Martin Luther King Day Optional film: Dr. King's 1963 speech "I Have a Dream"	Procedural recursion The Towers of Hanoi Graphical recursion **Read: Chapter 8**	Recursive backtracking Solving a maze **Read: Chapter 9**

Syllabus—Week 4

26 Backtracking and games The minimax algorithm **Read: Sections 9.2-9.3** **Due: HW #2 (Using ADTs)**	28 Algorithmic efficiency Big-O notation Sorting algorithms **Read: Chapter 10**	30 The C++ memory model Pointers **Read: Chapter 11** **Due: RandomWriter contest**

Syllabus—Week 5

February 2 Dynamic allocation **Read: Chapter 12.1-12.8**	4 The **`editor.h`** interface **Read: Sections 13.1-13.3** **Due: HW #3 (Recursion)**	6 Implementing editors **Read: Sections 13.4-13.5**

Syllabus—Week 6

9	11	13
Templates Implementing stacks	Implementing queues Implementing vectors	The **`StringMap`** class The idea of hashing Implementing **`HashMap`**
Read: Section 14.1-14.2	**Read: Sections 14.3-14.4**	**Read: Chapter 15** **Due: HW #4 (Boggle)**

Midterm Exam
Tuesday, February 10
2:15 or 7:00 P.M.

Syllabus—Week 7

16	18	20
President's Day (no class)	Binary search trees Balanced trees Implementing **`Map`** **Read: Sections 16.1-16.4** **Due: Recursion contest**	Sets in mathematics Implementing sets **Read: Sections 17.1-17.3**

Syllabus—Week 8

23	25	27
Graphs Standard traversals	Graph algorithms Shortest-path algorithms Minimum spanning trees	Inheritance in C++ Defining shape classes
Read: Sections 18.1-18.4 **Due: HW #5 (PQueue)**	**Read: Sections 18.5-18.6**	**Read: Sections 19.1-19.2**

Syllabus—Week 9

March 2	4	6
Expression trees Representing expressions **Read: Section 19.3**	Parsing strategies Overview of BASIC **Read: Section 19.4** **Due: HW #6 (MazeMaker)**	C++ in the real world Using the STL The mysteries of **`const`**

Syllabus—Week 10

9 Advanced algorithms Google's Page Rank DAWGs and lexicons Heaps and priority queues **Read: Sections 16.5, 18.7**	11 Strategies for iteration Function pointers Function objects The **`<algorithm>`** library **Read: Chapter 20**	13 Further adventures in CS (optional) **Due: HW #7 (BASIC)** **Due: BASIC contest**

Final Exam
Tuesday, March 17
8:30–11:30 A.M.

Assignments in CS 106B

- Assignments in CS 106B are due at 5:00P.M. Assignments that come in after 5:00 will be considered late.
- Everyone in CS 106B starts the quarter with two "late days" that you can use whenever you need some extra time. In my courses, late days correspond to class meetings, so that, if an assignment is due on Wednesday and you turn it in on Friday, that counts as ***one*** late day.
- Extensions must be approved by the TA (Kevin Miller).
- Assignments are graded by your section leader, who discusses your work in an interactive, one-on-one grading session.
- Each assignment is given two grades: one on functionality and one on programming style. **Style matters.** Companies in Silicon Valley expect Stanford graduates to understand how to write code that other programmers can maintain.

The CS 106B Grading Scale

- Functionality and style grades for the assignments use the following scale:

✚✚	A submission so good it “makes you weep.”
✚	Exceeds requirements.
✓✚	Satisfies all requirements of the assignment.
✓	Meets most requirements, but with some problems.
✓−	Has more serious problems.
−	Is even worse than that.
−−	Why did you turn this in?

Contests

- CS 106B will have three contests as follows:
  - The Random Writer Contest associated with Assignment #2
  - The Recursion Contest associated with Assignments #3 and #4
  - The BASIC Contest associated with Assignment #7
- First prize in the contest is a score of 100% on one of the graded components of the course, typically the final exam.
- As an additional incentive, entering any of the contests gives you chances to win an additional grand prize in a random drawing at the end of the quarter.
- Securing a runner-up prize or an honorable mention on any contest gives you additional chances in the random drawing, as does having an assignment submitted as a ++ candidate.

Honor Code Rules

Rule 1: You must indicate on your submission any assistance you received.

Rule 2: You must not share actual program code with other students.

Rule 3: You must not look at solutions posted on the web or from other years.

Rule 4: You must be prepared to explain any program code you submit.

The “Hello World” Program

One of the important influences on the design of Java was the C programming language, which was developed at Bell Labs in the early 1970s. The primary reference manual for C was written by Brian Kernighan and Dennis Ritchie.

On the first page of their book, the authors suggest that the first step in learning any language is to write a simple program that prints the message “hello, world” on the display. That advice remains sound today.

1.1 Getting Started

The only way to learn a new programming language is to write programs in it. The first program to write is the same for all languages:

Print the words

```
hello, world
```

This is the big hurdle; to leap over it you have to be able to create the program text somewhere, compile it, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.

In C, the program to print “**`hello, world`**” is

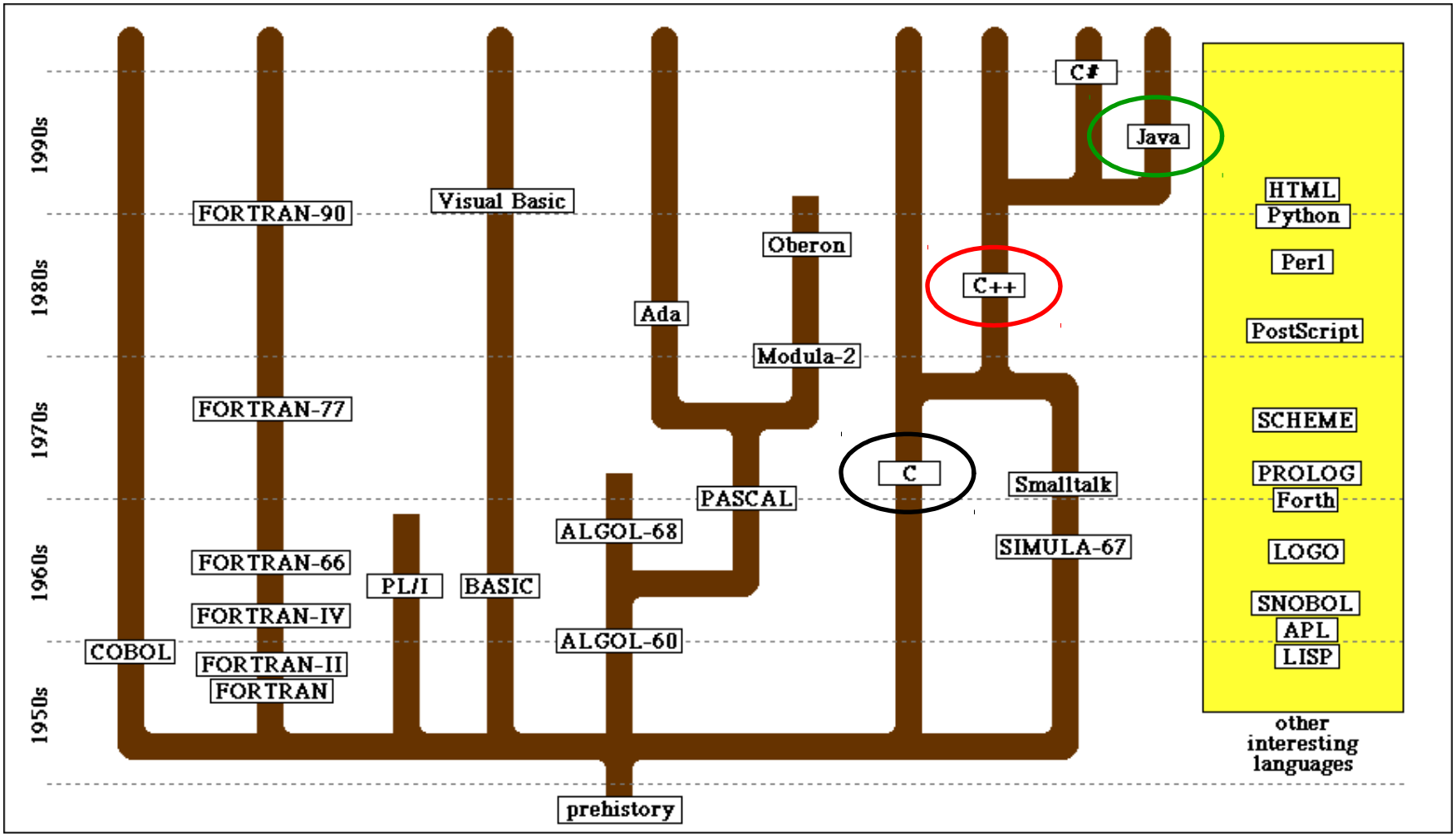
```
#include <stdio.h>

main() {
    printf("hello, world");
}
```

The “Hello World” Program

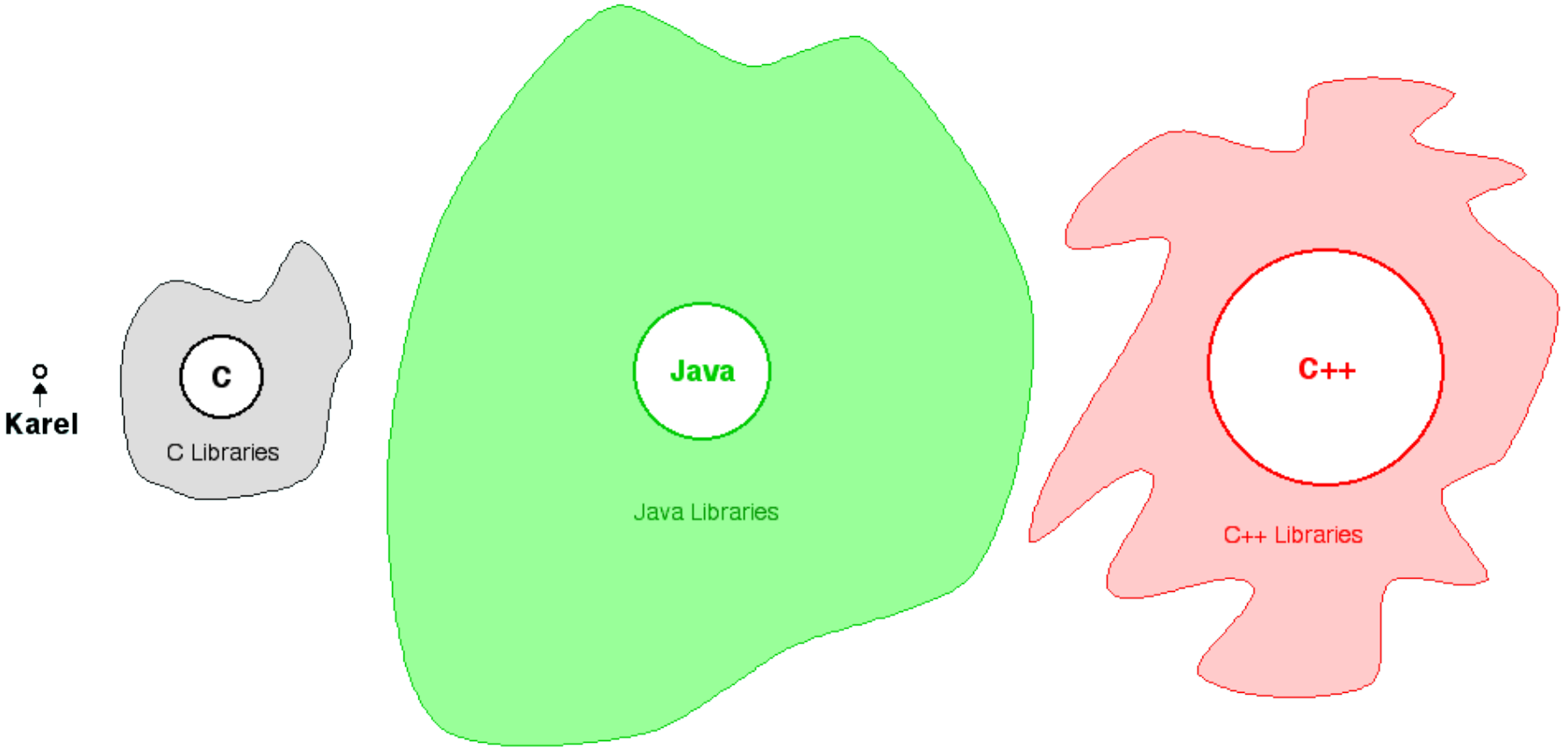
```
/*
 * File: HelloWorld.cpp
 * --------------------
 * This file is adapted from the example
 * on page 1 of Kernighan and Ritchie's
 * book The C Programming Language.
 */

#include <iostream>
using namespace std;

int main() {
   cout << "hello, world" << endl;
   return 0;
}
```

Download: HelloWorld.cpp

Evolution of Computer Languages



Size and Complexity in Languages

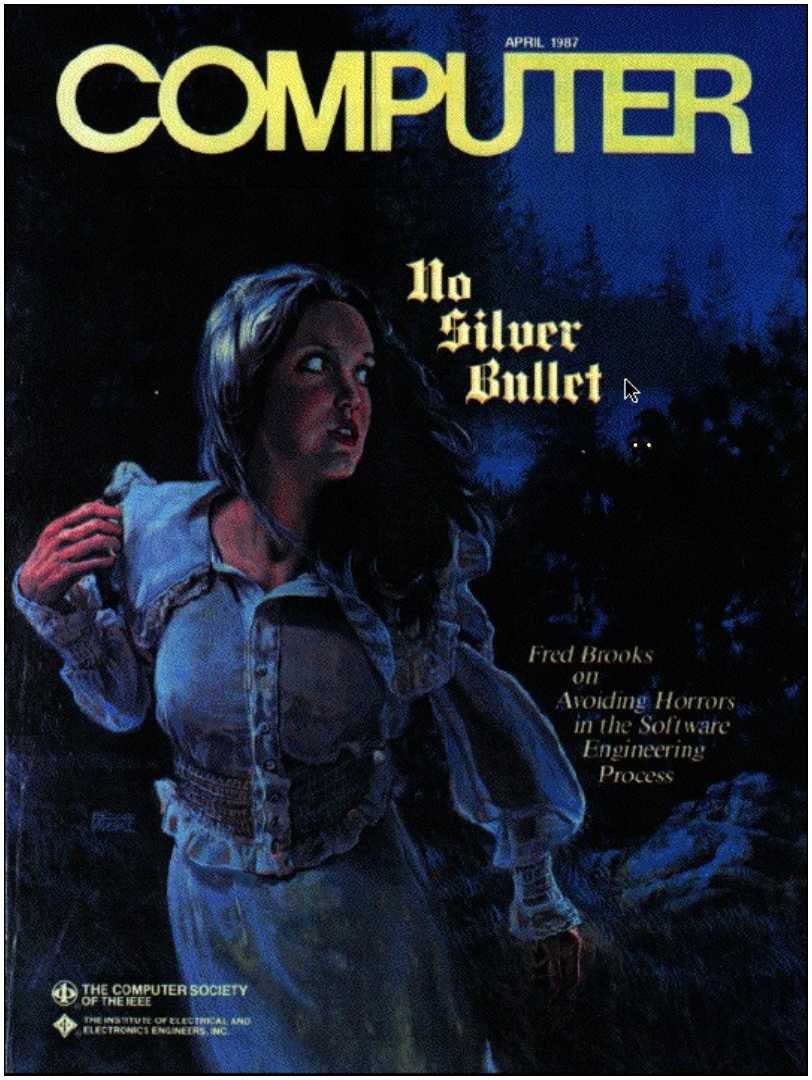


Essential and Accidental Complexity

To see what rate of progress one can expect in software technology, let us examine the difficulties of that technology. Following Aristotle, I divide them into *essence*, the difficulties inherent in the nature of software, and *accidents*, those difficulties that today attend its production but are not inherent. . . .

The complexity of software is an essential property not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence.

—Fred Brooks
"No Silver Bullet"
IEEE Computer, April 1987



C++ *vs.* Java: Accidental Differences

- The type of Boolean variables is **`bool`** instead of **`boolean`**.
- The type **`char`** represents an 8-bit ASCII character instead of a 16-bit Unicode character.
- Programs in C++ begin by calling a function named **`main`** that returns an integer status, which is 0 on successful completion.
- All functions used in a C++ program must be preceded by a ***prototype*** that defines their parameter structure.
- There are many accidental differences in the methods exported by the **`string`** class. For example, the second argument to the C++ **`substr`** method is the *length* and not the *ending position*.
- The discipline used for console I/O is significantly different, but not essentially so.

C++ *vs.* Java: Essential Differences

- C++ allows you to get very close to the inner workings of the machine. Java protects you from these details.
- C++ supports ***call by reference***.
- C++ allows programmers to modify the language in more ways than Java does. In particular, a C++ class can redefine the meaning of operators such as **+** or **<<**.
- C++ has no garbage collector of the sort that Java does. As a result, **you** are responsible for freeing any memory that is allocated on the heap.
- Objects in Java are always allocated in the heap. In C++, objects are typically stored on the stack to simplify the task of memory management.
- C++ supports ***multiple inheritance***, which means that classes can inherit behavior from more than one superclass.

The End