

Welcome to Computer Science  
A Cultural Introduction

23 January 2012  
Bruce Molay

### Welcome Aboard!

If this is your first course in computer science, this short introduction will show you the ropes and help prepare you for this journey. Just as life on the seas is different from life on dry land, studying and learning in the realm of computer science differs from studying and learning in other fields you may have visited.

In fact, computer science is a unique department here at Tufts. We are the only department that is part of both Arts and Sciences and also Engineering. Like a platypus, a course in computer science defies normal categories. Once you learn to deal with it on its terms, though, you will do fine.

### What Should I Expect from a CS Course?

An introductory computer science course is like a language course or a math course, but with a surprising twist.

First, you learn by doing. Just as in a language course or a math course, you have to learn new vocabulary, new syntax, new idioms, new ways of describing things you know how to say in other terms. Each of these skills builds on the ones before. This means regular practice, drill, repetition. Simply reading a book on Japanese grammar does not make you a fluent speaker. You have to do it. A lot. If that sounds unpleasant, you probably want to get back on dry land now.

Second, at the same time you are learning basic vocabulary and grammar, you will have to use them to write compositions. Here, we call them programs. Most of you learned expository writing in middle school and refined that skill in high school. Given a topic, you have to construct an answer using facts, ideas, arguments, and logic. By the time you got to college, your professors could expect you to know how to build those compositions: making note cards, writing an outline, using topic sentences, organizing presentation of facts and explanations of principles. Here, you have to develop similar skills in a new, highly restricted language. Sort of like having to write an International Relations paper on how to manage a tricky political situation, but being limited

to sonnet form using only words that have three syllables.

And now for the twist. In every other course, once you write the last paragraph of your paper, prove the last theorem, solve the last problem on the problem set, the assignment is mostly done. Maybe some proofreading, editing, double checking your calculations.

In computer science things are different. Once you finish typing in the last part of your computer program, your work is just beginning. This fact is well-known in the field as the *90-90 rule*: "The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time."\*

### How Should I Adapt to this New Field?

There are only three basic rules to learning effectively in a computer science course, and these rules address the three items mentioned in the previous section.

First, treat this like a learning a language or developing a physical skill and practice every day. It does not have to be a lot, but, like going to a language lab, reviewing flash cards, or working out in the gym daily, steady, consistent practice is the only way to build your skills and muscles. Running a half hour each day for a week is not too stressful, and it produces real results. Procrastinating, then running for three and a half hours at the end of the week is not even possible.

Second, work on developing skills in this new form of composition. People have different, personal, successful approaches to planning and writing papers and essays. Some use note cards, some make outlines, some write lots of paragraphs then rearrange them. You have your own way that has gotten you this far in education. For this course, you have to start again, and find a

---

\* Attributed to Tom Cargill of Bell Labs in Jon Bentley's September 1985 "Programming Pearls" column in Communications of the ACM, in which it was titled the "Rule of Credibility".

way to apply new skills to compose little facts and ideas into solutions to problems. Different people have different approaches, but everyone needs to work on developing this skill. Write outlines, draw diagrams, explain your ideas in comments before and as you build, try explaining your plan to a dog.

Finally, never forget the 90-90 rule. Most programs you write will have bugs. Some might not, but never count on that. One strategy used by the pros is to build small parts of the program and test them right away. Then fix those problems. Then build the next little part, and test it, then fix those problems. You have to spot the problems and fix them sooner or later. Sooner is better than later, because the problems are smaller and easier to spot. Even if you plan well, expect bugs. Learn techniques for isolating the problem. You need to combine the objective eye of an ace proof-reader and the analytic mind of a private detective.

### **Enjoy the Trip**

Some college courses are advanced versions of topics you studied in high school and before. Math, you've done that; science, been there; English, every year; History, reading, papers, facts and ideas. But computer science for most people is something completely different. Sticking with your approaches from other fields is like taking a semester abroad but trying to rely on habits and perspectives that work back home. Be prepared for a new experience, learn the ropes, and bon voyage.