Aiding Self-motivation with Readings in Introductory Computing

Gary T. Leavens
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports

Part of the Computer Sciences Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/98

This Article is brought to you for free and open access by the Computer Science at Digital Repository @ Iowa State University. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Digital Repository @ Iowa State University. For more information, please contact digirep@iastate.edu.
Aiding Self-motivation with Readings in Introductory Computing

TR94-08a
Gary T. Leavens

October 3, 1994

Iowa State University of Science and Technology
Department of Computer Science
226 Atanasoff
Ames, IA 50011
Aiding Self-motivation with Readings in Introductory Computing

Gary T. Leavens
TR #94-08a
May 1994, revised October 1994

Keywords: motivation, readings, bibliography, introductory computer science, education.


© Gary T. Leavens, 1994. All rights reserved. This paper has been submitted for publication, so the copyright will be eventually assumed by the publisher of the final version.
Aiding Self-motivation with Readings in Introductory Computing

Gary T. Leavens
Department of Computer Science, Iowa State University
Ames, Iowa 50011-1040 USA leavens@cs.iastate.edu

October 3, 1994

Abstract
Students can achieve self-motivation and a broader appreciation of computing by reading widely about computing. This paper advocates discussing self-motivation with students, and suggesting that they read widely as a means to that end. A discussion of how to present these ideas effectively, and an annotated list of suggested readings, appropriate for undergraduate majors in computing, are included.

1 Introduction
Since a student's motivation and interest are important factors in how effectively they learn, motivating students and capturing their interest is vital. When teaching majors, students who plan to work in the field, increasing their motivation and interest is doubly important, as it can have long-term positive effects.

This paper makes two contributions. First, it advocates discussing self-motivation and its importance with majors. Second it suggests a concrete way to actively encourage students to promote their own self-motivation and interest, and offers specific reading suggestions appropriate for majors who are starting their study of computing.

An incidental feature of these reading suggestions is that they can also be an aid to teachers who are looking for examples that give students an appreciation of how computing ideas are used in practice and what being a computing professional is like. However, such reading suggestions will do even more good if these books move off the teacher's shelf and into the hands of their students.

2 Reasons for Advocating Wide Reading in Computing
My professional career in computing really started at the library. It may be that my first professors encouraged me to read books about programming, but in any case I was often in the library, or reading a book that had nothing in particular to do with the class I was taking. My reading had several beneficial effects. It kindled my interest in computing, it gave me good background for my classes, and it motivated me to learn more. Most computing professionals I know avidly read about computing and read more widely when they were students.

When I first started reading about computing, in the early 1970s, there were some books about FORTRAN and IBM JCL on the shelves, but the computing section of the library
was not yet drowning in books about Pascal (let alone, C, C++, DOS, and Windows). In the 1970s, even the bookstores featured mostly what would now be called “professional computing” books (like Knuth’s *The Art of Computer Programming*).

Today, while the shelves of the university library still have many professional computing books, the density of such books on the shelves has surely been decreased by all of the books about Pascal, C, C++, DOS, Windows, etc. And nothing need be said about what kind of books are on the shelves in the computing section of most bookstores and public libraries.

While there is nothing wrong with this wealth of practical books about popular computer languages and systems, it poses a problem for the college student new to computing. Such a new student is in danger of seeing the field of computing as simply a very large set of skills. The danger is that a new student will not see that computing has intellectual interest, problems of importance, ideas that have wide applicability, and a literature that is worth reading. In short, even students who already have good reading habits may need to be guided to the intellectual heart of computing if they are to see computing for what it really is, and if their reading is to increase their motivation and interest in the subject matter taught at the college level.

One way to guide students to the intellectual heart of computing, and thus indirectly guide their reading, is to survey computing as a whole in the first class in the curriculum [9] [8] [10]. However, there are many, myself included, who do not favor such an approach. So for various reasons, many schools will still be teaching a first course in computing which is largely a programming course. Such a course, can also help guide students to the intellectual heart of computing if one presents many problems as “word problems,” stories that give context to the computing exercise. This context helps students to see what computers are good for, and to see more of what being a computing professional is like. Abelson and Sussman’s classic book is a good example of such an approach [1].

Properly done, a survey course or a first programming course will motivate and interest students, and give them the background they need for further study. However, one would still wish to speed up student’s development and to increase their motivation, by giving them more explicit guidance and encouragement in their reading.

There are several approaches to discuss self-motivation and present such an introduction to the computing literature to students. One way is to simply present a list of introductory readings to students as a handout. Some students will ignore it, and most will be too busy during the school year to do much reading outside class; nevertheless, they may turn to it in the summer or in later years. (Thus it is important that remarks about the list be included with the list itself.) However, for a few students, even this approach will be very helpful; these students are global learners [3, 4], who have difficulty learning a subject if they do not see its context and motivation. By passing out a list of introductory readings, one gives global learners some of the tools they need to succeed. Some other students, although planning a career in computing, have not worked with computers before, and will be grateful to be pointed in the direction of literature that will help them get more perspective on the field. At the opposite extreme are those students who are immersed in the details of computers; discussing computing literature with them is a small step towards showing them the intellectual heart of computing, which they will need to understand and appreciate if they are not to drop out of college study in frustration.

However, other ways of discussing self-motivation and introducing computing literature would be more effective. Devoting some class time is a step in the right direction. More effective yet would be some homework project. For example, assigning a book report as a term project. Although there is a danger in promoting self-motivation with required work,
allowing the student to choose his or her own reading material should assuage some of that
problem. Making reading an assignment further emphasizes the importance of something
that, in the long run, must be the student’s own responsibility.

3 Introducing Self-Motivation through Wide Reading to Students

In discussing self-motivation and the use of wide reading to promote it, I think it best to be
frank with students, and to address their concerns. These days, many if not most students
seem to be motivated to enter computing because they want to get a good job. They seek
skills, not intellectual stimulation for its own sake. But such an attitude is short-sighted.
Majors in computing should be reminded that, if they stay in their chosen field during their
working years, they may be working in computing for thirty or forty years. Students also
underestimate their own capacity and need for intellectual stimulation. So I tell students
that there is more to life than getting a job and working, and there is more to being a
computing professional than having a diploma, even more than having the needed skills.
Part of this “more” is enjoyment and intellectual stimulation. This may sound corny, but
I go on to tell them that if they enjoy their work, they will do better at it. Furthermore, I
tell them that this process feeds on itself: as they improve they will enjoy their work more,
and because of that they will want to learn more.

Leaving aside the Socratic dilemma of how to get the process going, I tell students
that it is important for them to find an enjoyable way to increase their performance and
enjoyment. One answer is given by the many gifted authors who ready to help them.
These true teachers have found ways to write about their subject in a way that is fun and
enlightening.

I conclude with a caveat: while reading such books has helped me enjoy and learn
computing, there are many ways to learn a subject. Students should not despair if this
approach does not work for them, or even if they find they do not like computing after all.
The only test for whether a student should work in computing or any other field is whether
they do well at it and enjoy it. The important thing is to find some way to increase their
skill and enjoyment.

4 Computing Literature Recommendations

The first recommendation for the students should not be any particular book, but the
library itself. Tell the students exactly where in the library the computing books may be
found—give the call numbers. Since students may have interests different one’s own, it
helps to look around a bit to get a suitably wide range of call numbers. For example, in
my university’s library, some computer engineering literature is found in a different section
from the computer science literature. Also note where reference materials and current
periodicals are kept. Point out that it takes time to get acquainted with what is in the
library, and that the list you are presenting only discusses a small sample of the available
literature. Therefore encourage students to browse in these areas, perhaps by giving them
an assignment to write down two or three titles that are of interest to them personally, and
to state the reasons why they look interesting. Also encourage students to ignore anything
that is not of interest or that looks too technical (at first).
The following are some of the recommendations I have made to majors in computing taking a first programming course. The readings are divided by topic. In addition to the literature discussed below, I also recommend books about the specific programming language used in class, which is Scheme [7, 1, 5, 2, 6]. These books also contain many interesting programming ideas. I also recommend books about the operating system and editor that the students use, although there is little need to discuss those books here. Finally, if the course is not a survey of computing, it is good to recommend such a book, for example [10]. If nothing else, such a book will be an aid to global learners.

I also recommend that students may want to read books about science or some other science in general to get more perspective. For example, I tell students that Stephen Jay Gould writes wonderfully about biology, and Issac Asimov writes well about most sciences.

4.1 Interesting Reading about Computing

The following two books by Douglas Hofstadter are classics with a very eclectic mix of philosophy, art, and computer science. These should be of interest to all students of computing. The second is an easier read than the first, since the second is a collection of shorter essays.


A. K. Dewdney has written several popular books on computing. Highly recommended as in introduction to the field is his *The Turing Omnibus*, which is a collection of essays. His book, *The Magic Machine* features lots of “programming recreations”. His most recent book is *The Tinkertoy Computer and other machinations*, which is also a collection of easily approachable essays.


Computer engineers should not miss reading Tracy Kidder’s account of the development of a new (at the time) Data General minicomputer. It is also good for giving (anyone) some idea of what it’s like working on a major project.


The following two books are at opposite ends of the spectrum in introducing computing hardware. For a very basic introduction to the hardware and software in a personal computer, read White’s book *How Computers Work*. An excellent technical introduction to the engineering side of computing is Ward and Halstead’s book.


For some amazing computer graphics, check out the following books by Clifford Pickover.


The following book by Poundstone is a nontechnical account of how simple computational processes (such as Conway’s game of life) relate to the limits of scientific knowledge. It makes fascinating reading and is a good way to connect computing to science at large.


Artificial intelligence is the subject of the following books. William Robinson’s book is an interesting philosophical discussion about the possibility of Artificial Intelligence. Daniel Crevier provides a fascinating historical discussion in his book.


History is the subject of the next two books. Hodges book is a good biography about Alan Turing, one of the pioneers of computing. The history of programming languages is always interesting to students, and is the subject of the preprints edited by Wexelblat.


The following books are about more theoretical topics. Information theory is closely related to computing and is the subject of R. W. Lucky’s book. David Harel’s book may be a bit advanced, but gives a good overview of Computer Science.


4.2 Readings about Programming in General

Technical books specifically about programming may be hard for new students to grasp. However there are some things to recommend.

One place to start is with works that discuss the outlook of a master programmer. A good example is Dijkstra’s classic lectures, *The Humble Programmer*, which should be of help to even beginning students. It is short enough it could be assigned as a reading for a whole class. The classic book by Brooks listed below discusses the problems of large-scale programming and software engineering that concern programmers. On a smaller scale, Kernighan and Plauger’s book has many good guidelines.


More technical, but still accessible to students, are the following books by Jon Bentley. The *Programming Pearls* books are collections of essays The book *Writing Efficient Programs* is more advanced, but will be of immediate interest to students who are fascinated by programming, but has much good advice, and will show them the importance of algorithms and data structures.


Finally, a list of readings about programming should contain some introduction to the more mathematical approaches to programming and program correctness. An excellent choice such as Dijkstra’s elegantly written book. A more recent and more introductory book is Cohen’s. It also features a calculational approach to predicate calculus.


4.3 Magazines and Journals on Computing

Several magazines cater to the home computer enthusiast; some of these, such as *Byte*, even feature an occasional article about programming (among the advertisements). But to read about programming and other computing topics, students should try some of the following
journals: Computer (published by the IEEE Computer Society), IEEE Software, Communications of the ACM (also known as CACM), and ACM Computing Surveys. Sometimes IEEE Transactions on Education and ACM SIGCSE Bulletin have good introductory articles as well. These are available in the library, but students should also be urged to subscribe by joining the ACM or the IEEE Computer Society or both. Tell the students where and how to join, and whether there are local meetings of student chapters.

5 Thinking and Problem Solving Recommendations

Some students have trouble in programming because they have poor problem solving skills. The introductory class should help sharpen those skills, but occasionally students need more help. For such students, a good recommendation is Polya’s classic How to Solve It. This book has been read as long as people have been programming (at least with high-level languages!). Mason’s book describes the problem solving process and gives many examples. Some colleges have a pre-computing course based on Mason’s book for students who want or need help in problem solving.¹


Students might also want to explore logic. Gries and Schneider’s book is a good introduction, and features a calculational approach that is well-suited to the needs of programming and careful reasoning. But more fun are the books by Smullyan, such as What is the Name of this Book?.


6 Conclusion

Teachers of introductory programming courses in computing can help students be self-motivated by explicitly discussing self-motivation, and by actively encouraging students to read widely in computing. Giving students guidance to the literature helps guide them to the intellectual heart of computing. Such guidance also helps students, especially global learners, by providing context. It also helps students decide for themselves whether they find the intellectual side of computing interesting and enjoyable. If they read some of the literature recommended above, they will.

¹For example, Emily Moore runs such a course at Grinnell College, Grinnell, Iowa.
Acknowledgements

Thanks to Charles Wright and to the faculty in my department for help compiling the initial list. Thanks to the anonymous referees for suggestions that improved the clarity and content of this paper.

References


Tech Report: TR94-08a
Submission Date: October 3, 1994