Computer Science/Informatics: The Study of Information World

Viera K. Proulx
College of Computer Science
Northeastern University, Boston, MA 02115, USA
vkp@ccs.neu.edu

Abstract
In this paper we argue that every secondary school curriculum should include a course of study of informatics/computer science. (In the USA the subject is typically called computer science, while the term informatics corresponds more closely to words used in other languages, as well as usage in other English speaking countries.) We present the highlights of the Model High School Computer Science Curriculum that has been endorsed by the Association for Computing Machinery [1] and give arguments for including a study of basic topics of informatics in every 'computer literacy' curriculum - be it at the elementary level, or in a course for university students not majoring in informatics. Finally, we discuss possible ways in which the teaching of informatics can be improved by using computer as a teaching tool, not just the subject of study.

Keywords:
Informatics as Study Topic, Curriculum Policies, National Policies, Pedagogy

Introduction

The world of informatics affects every aspect of today's civilized world. To make sure today's students grow up to be knowledgeable and informed citizens we need to include in the secondary school curriculum a study of basic underlying concepts of computers science/informatics. The pedagogy should resemble that of good science classes - with experiments, modeling, analysis and exploration of fundamental concepts.

What is Information World?

Students in high school study the Living World in their biology courses. They learn about the cells, their growth, they learn about the simplest organisms and their life cycle, until, towards the end of the course, they understand the general principles of the highest living organisms. They know something about genetics, they are aware of the impact all living creatures have on the ecology, and what is their place in the whole system of living things. They may never study biology again, but for the rest of their lives they will be informed citizens, able to follow the latest developments as reported in daily and weekly press, able to reason about the environmental impact of proposed projects, and they will learn to alter their behavior to minimize damage to the natural resources.

We could give similar argument about reasons why students take courses in physics and chemistry (to study the physical world or the 'substance of matter' world). The main point is that students learn about the basic components of these worlds, learn the basic laws governing their behavior, learn about the issues confronting scientists today, and learn how changes in these worlds can impact our daily lives.

The world of informatics is new. Before the invention of networked computers, access to all information has been tightly controlled. Information was stored in books and magazines, pictures, on movie reels, as art objects scattered throughout the world in museums, or it was stored in vaults of government documents. Before one could access such information, one had to perform tedious search, physically copy or borrow the document in question and return the copy after use. In the new world of informatics, we can access information directly, without an intermediary, we can perform search of a large number of documents, make copies, alter documents, or generate new ones. No longer do we need to wait for publishers, to make our results available to the wide public. It is not only a world we can observe and learn about, - it is a world we can manipulate and participate in.

What Do We Need To Know About the Information World?

Many countries require some aspects of informatics or computer literacy as a part of their secondary school curriculum. Students in China, Japan, Russia, Great Britain, The Netherlands, Malaysia and many other countries are required to learn something about computers, though specific requirements vary. Recently, UNESCO [11] published Informatics for Secondary Education: A Curriculum for Schools, to provide guidelines for creating national computing curricula. In the USA, the Association for Computing Machinery published The ACM Model High School Computer Science Curriculum [1]. Rather than focusing on computer 'literacy' ACM report calls for a high school computer science course on par with other sciences - chemistry, biology and physics, that are now a common part of high school curricula. The report identifies six core areas: Algorithms, Programming languages, Computer architecture, Operating systems and user support, Social, ethical and professional context, and Applications. In addition, the report recommends that one or more additional topics be studied in some depth - depending on the interest and abilities of the instructor. An appendix entitled *Scope of proficiencies of students successfully completing High School Computer Science Curriculum* serves as a guideline for teachers planning to teach such course.

We will examine the six core areas identified in this report and show how each area contributes to student's understanding of the information world.

Algorithms:

Algorithms have been invented and used for centuries, even millennia. With the advent of stored program computer, they became an independent subject of study. They describe the process that needs to be followed to achieve a desired goal. The basic building blocks - repetition, decision, encapsulation, the notion of initial conditions, input, and the resulting output, describe processes performed daily by scientists, clerks, doctors, engineers, - people in all walks of life. All students need to learn how to formulate the description of a process as an algorithm. When describing an algorithm student has to be careful to cover all possibilities, express the instructions in precise terms using a well-defined language (pseudocode, programming language, script). Lessons learned here transfer to many other fields and provide a foundation for logical reasoning.

Another aspect of study of algorithms is the complexity. Looking at the incredible powers of computers today, nothing seems impossible. Many laymen wonder why do we need yet bigger and faster computers. There are many simple examples that can be used to illustrate the time complexity, space complexity, problems that are intractable even though we can describe the algorithm. Biermann [2] shows how we can explain the notion of unsolvability to an ordinary person. By talking about these issues, seeing the scale of growth of the problem solution relative to the input size, students begin to understand the need for growth in computer size and speed, the need for working on new ways of solving problems, the impact of computing on our world.

The third aspect, which should form a recurring theme throughout the course, is the issue of abstraction and encapsulation. Computer systems are incredibly complex. Each layer (from hardware to different layers of software) provides the user with a new functionality and hides the implementation details behind an impenetrable shell. This principle of abstraction is a model of problem solving strategy that can be employed in solving many of the complex problems facing our society today.

The techniques used for representing algorithms - special languages, flowcharts, transition diagrams, pictographs - again are relevant in other situations as well.

Programming Languages:

Every literate user of a computer needs to learn at least one way of communicating with the computer. It may be a general purpose application program (like database, spreadsheet, word processor, statistical package), a menu driven system designed for the customer (e.g. hospital admission program, airline reservation system), or just plain higher level programming language (C, Pascal, FORTRAN, C++, Scheme, *etc.*). But without the knowledge of basic principles of programming - levels of programming languages (from machine level to the highest applications), understanding of the compilation or translation process, the layers of abstraction encapsulated by different languages - users of computers cannot understand why certain features work the way they do, they are not able to customize the application to fit their needs, and have a much more difficult time learning how to use new applications. By understanding what is behind the buttons that are pushed, key strokes being typed, users gain ownership of the system they work with. By knowing what is involved in the translation or compilation, users understand the need for being 100% correct in their instructions, and can guess how the computer interpreted a faulty command.

Examples of simple machine language, a simple finite state automaton that recognizes a given string, simple model of compiler - all can be employed to illustrate the principles used in designing programming languages and their compilers.

Computer Architecture:

The understanding of the language computer ‘speaks’ goes hand in hand with understanding what the computer is made of. There are numerous analogies to the simple model of computer: input/output devices, a CPU and a memory that stores data and instructions. A restaurant kitchen, with cook acting as a CPU, with recipes stored in his head or on a paper, ingredients being data needed by the program, with waiter bringing in the orders and carrying the results to the customers, is just one of them. Description of a knitting pattern for a pullover is a program executed by the CPU consisting of needles, and hands and eyes of the knitter. The input is a ball of yarn, the output is a finished piece of garment.

To understand that computer instructions can be encoded together with the data used by the program, students need to learn about the various ways computer can represent the data. Starting with bits and bytes as basic components of information students learn how different interpretations can assign meaning to a collection of bytes. Is this important? Well, humans have been using different encoding schemes for ages - from the old hieroglyphs, to modern alphabets and language representation systems, Arabic and Roman numerals, Morse code, telephone numbers, addresses and postal codes, bar codes on nearly all items sold, traffic signs, airport signs identifying different services. The world is full of symbols representing different objects and ideas. By looking on the ways in which computer represents information, students learn about the limitations of different representations - and at the same time - about the limitations of computer systems.

Studying the various ways of representing data and information (including sound, pictures, speech, video), studying the limitations of these representations, looking at the ways one can be transformed into another - students gain a deeper understanding of the world around them: the video recording, sound recording, data transfer over electronic networks, the ability to manipulate and edit data, ability to create new sounds and images, ability to organize data into knowledge bases used interactively over large distances.

Operating Systems and User Support:

One of the major stumbling blocks for a new user of a computer system is learning the language used to communicate with that particular system. “How comes I cannot use the PC, even though I have no problem with my Mac?” is the type of question users of computers are asking every day. Once students understand how computers communicate with different peripheral and input/output devices, when they learn how is the data stored on them, the mystery of the operating system interface starts lifting. It is important to realize, that all computer systems provide essentially the same functionality and that the user interface is just a front end allowing the user to request specific services.

Students should understand that the operating system is the manager of all computer system resources - it keeps track of everything we are ever going to look for, it organizes this information into tables, directories, and lists, so that any legitimate request we will make will receive a correct response. Again, one can draw parallels to complex systems from daily life - students managed by teachers, managed by school principals, managed by the superintendent, managed by the school board, overseen by State Curriculum Boards, etc. One can compare it to a large organization, or to a living organism. Our interface to the outside world allows us to receive inputs through touch, smell, sight, hearing, and taste - we react to these, generating other new sensations. Again, we explore the levels of abstraction and the concept of information hiding.

At this point students should also learn how to communicate with other computers worldwide - using different network services. It helps them in understanding both the power of computers today, and the dangers associated with their use.

<u>Social, Ethical and Professional Context:</u>

Today, computers affect almost every aspect of our lives. Computers keep track of business transactions, medical examinations and treatment, our educational achievements, record the census data, stock market transactions, assist in all banking transactions. They allow us to talk to a friend or colleague thousand of miles away, look at pictures from Jupiter, send a robot to the depths of the sea, or to the moon. We can look up instantaneously a topic in an encyclopedia, read the latest news from the news wire, or download from the network numerous free programs written by people all over the world. The way people communicate and conduct business is changing is a revolutionary way that can only be compared to the impact of the invention of the printing press.

The unimaginable growth of computer usage both in volume and in its scope is bringing with it legal and ethical problems not encountered before. Even non-users of computers needs to be aware of dangers of potential misuse of records that different businesses keep about them. A manager implementing a new computer system for a business must understand the risks involved - viruses, software bugs, breaches of system security from outside or inside, liability in the event of failure, *etc*. The society is just introducing new laws designed to protect the rights of computer user, computer software writer, or a private citizen whose name and data is stored in one of a myriad data bases.

Another problem facing today's society is the issue of equity. More affluent segment of society is gaining an even greater advantage from being literate users of computers, leaving poor and uneducated class even further behind. While wealthier school district provide a number of different computers for their students, poor school districts are having a hard time to enter the computer age. Every secondary school student should be required to think about these issues, the problems, the solutions, the impact, and the goals for the future.

<u>Applications:</u>

It has been said many times, "you cannot learn to ride a bicycle by reading about it". Similarly, to learn about computers, one should have a reason for using them, and should learn how to use them while learning about them. The ACM report recommends that all students learn how to use at least a couple of typical computer application programs. Why is it not sufficient to become a skilled computer programmer? Programming languages such as Pascal, Algol, C, Ada or Scheme operate at quite low level compared with the complex application programs available today. It is important to teach students to use the right tool for the job at hand. By learning to use more powerful tools students gain deeper appreciation of the power of computers. A spreadsheet program can evaluate formulas, compute statistics and present the numerical information in a wide variety of graph formats. When used skillfully, it can even generate an animation or solve an equation. Some of the applications in general use are real gems, but students won't find them, unless they learn to explore them..

Students should also learn about combining several application when trying to solve a problem - creating a file of data extracted from a data base, using this file to create a spreadsheet and plot the data, copy the completed chart into a report produced with a word processor. Besides training a powerful computer user, this type of exercise reinforces the message about all data being basically the same - with the differences being only in the interpretation. For classes where the main goal is 'computer literacy', studying several application programs, including the embedded macro language provides a powerful basis for discussion and presentation of computer science topics outlined in this paper. The ACM report includes several model implementations - one of them is based on teaching applications.

Additional Topics:
Today computers are used in most unexpected places. New uses are appearing every day. For students to appreciate the potential growth and impact of computing on our daily lives, they should see one or more examples in which computer is used to help solve new types of problems. The main driving force here should be teacher's enthusiasm for a particular way a computer can be used. Students can write music, learn about image processing, explore graphics programs and write a small part of one, they can experiment with sound and speech analysis or look at compression algorithms and compare their behavior.

In conclusion, by studying the core topics of computer science, students in secondary school will receive a solid foundation and understanding that will allow them to be literate users of computers for years to come, independently of the types of computer systems, or that nature of the computer application they will be asked to deal with. The computer will become a powerful tool they will use daily to improve their efficiency and productivity, as well as to gain access to new information and knowledge. Also, students already focused on understanding of computer science topics will broaden their appreciation for the power of computers by learning to use programs and applications written by others, and making them their daily work tools.

How About Methodology?

Computers, and especially their ability to combine sound, pictures, text, and video into a hypertext system with multitude of links, searching options, and avenues for exploration, is changing the way many subjects are taught. Many students learn about a particular topic by creating their own multimedia presentations or tutorial systems. Browsing through one of the exhibits at a major educational computer conference, one is overwhelmed by the various options available to teachers and students today - unless one is looking for products that support teaching of computer science topics. The shoemaker goes barefoot.

If we want to make the study of informatics accessible to all students, we need to think carefully about the methodology, course materials, and supporting computer based educational software that is to be used in such courses. What should be our goals? We present some examples - most are just flights of fancy - hopes and dreams, while some parts exist in various forms and can be obtained from private or commercial sources. We need to do more such work, and find ways to share them with teachers everywhere.

To study computer architecture, student should be able to play with a model that allows them to feed in the input, examine the memory, see the bits going through adder and other logic circuit, see the program that is being executed, explore what happens if some of the parameters, statements, or inputs are changed. The processes happening inside of the computer are very complex, and many students have a hard time visualizing or understanding this dynamic behavior. When learning about some algorithm, one can show the various states on the blackboard, or one can use a computer model that students can experiment with and control on their own [us, Stasko]. When learning about algorithm complexity, students can perform actual timing algorithms using implementation written by others, collect the timing data and use a spreadsheet program to analyze and display the results [5]. A multimedia presentation guiding student through the history of computing, showing the types of problems different generations were capable of solving, displaying the diagrams and photographs of different systems would make the subject come alive. It would also help students to develop deeper appreciation for the incredible growth of computer industry over the past few decades. A model of finite state machine would turn the explanation of simple language recognition systems into an interesting game. A visual representation of a neural net system and the changes it goes through during the training phase is another example. Program animating some of the graphics algorithms - showing simultaneously the computations being done, the desired goal for the drawing, and the progress in creating the drawing is another example. Or we may want to see a graphical representation of the different files used to implement a particular data base, the links between them, and an animation of the processing of a given query.

Computer Science professionals should become a leading force in developing high quality computer based materials for teaching computer science concepts to students at all levels.

Bibliography

1. ACM Task Force (1993) *ACM Model High School Computer Science Curriculum: Report of the Task Force on High School Curriculum of the ACM Pre-College Committee*. ACM Press, New York.

2. Biermann, A. W., (1990) *Great Ideas in Computer Science: A Gentle Introduction*, MIT Press, Cambridge, MA, USA.

3. Brown, C., Fell, H. J., Proulx, V. K. and Rasala, R., (1992) *Using Visual Feedback and Model Programs in Introductory Computer Science*. Journal of Computing in Higher Education, **4**(1), 3-26.

4. Brown, C., Fell, H. J., Proulx, V. K. and Rasala, R., (1992) *Programming by Example and Experimentation*, in Computer Assisted Learning, Proceedings of the 4th International Conference on Computers and Learning, ICCAL '92, Wolfville, Nova Scotia, Canada, June 1992, (ed. I. Tomek), Springer Verlag, pp. 136-147.

5. Rasala, R., Proulx, V. K. and Fell, H. J., (1994) *From Animation to Analysis in Introductory Computer Science*, in Proceedings of ACM Computer Science Conference, Phoenix, AZ, March 1994, pp. 61-65.

6. Stasko, J. T., (1990) Tango: *A Framework and System for Algorithm Animation*, IEEE Computer, **23**(3), 27-39.

7. Taylor, H. G., Aiken, R. M. and van Weert, T. J., (1993), *Informatics Education in Secondary Schools*, in IFIP Working Group 3.1 Guidelines for Good Practice.

8. WG 3.1, (1991) *Proceedings of the IFIP WG 3.1 Working Conference "The Impact of Informatics on Organization of Education"*, Santa Barbara, California, August 1991, Elsevier Scientific Publishers.

9. WG 3.1, (1993) *Proceedings of the IFIP WG 3.1 Open Conference "Informatics and Changes in Learning"*, Gmunden, Austria, June 1993, Elsevier Scientific Publishers.

10. WG 3.1, (1994) *Proceedings of the IFIP WG 3.1 Working Conference "Integrating Information Technology into Education"*, Barcelona, Catalonia, Spain, October 1994, Elsevier Scientific Publishers.

11. UNESCO, (1994), *Informatics for Secondary Education - A Curriculum for Schools*, Produced by a working party of the IFIP, UNESCO, Paris.