FOUNDATIONS OF COMPUTER SCIENCE

M. S. Carberry, H. Khalil,
J. Leathrum, and L. Levy

COMPUTER SCIENCE PRESS

FOUNDATIONS OF COMPUTER SCIENCE

M. S. CARBERRY
H. M. KHALIL
J. F. LEATHRUM
L. S. LEVY

All of the authors are on the faculty of the University of Delaware

COMPUTER SCIENCE PRESS, INC.

Computer Science Press, Inc.
11 Taft Court
Rockville, Maryland 20850

1 2 3 4 86 85 84 83

Library of Congress Cataloging in Publication Data
Main entry under title:
Foundations of computer science.

Bibliography: p.
Includes index.
1. Electronic data processing. 2. Electronic digital computer. I. Carberry, M. S. II. Series
QA76.F6373 001.6'4 78-27891
ISBN 0-914894-84-6

First paperback printing 1983
First hardcover printing 1979
Second hardcover printing 1982

Preface

Computer science is a young field which is undergoing rapid change compared to more established academic disciplines. This movement has led to a variety of curricular approaches. Moreover, the different environments in which computer science has grown-up, and the sundry research interests of faculties has encouraged different courses of study, and attempts to define the nature of the science—which is surely a source of confusion to other disciplines.

It is a truism, that like the five blind Indian sages encountering the elephant, different computer scientists have come to different definitions of computer science. Our (first order) definition is that *computer science is the study of the theory and practice of programming computers.* This differs from the most widely used definition by emphasizing programming as the central notion and algorithms as a main theoretical notion supporting programming. The study of heuristic programming is certainly part of our computer science. While it is premature to measure the relative significance of empirical methods in our science, ignoring them would give a distorted view of the science.

This text was designed for the student taking his or her first course in Computer Science. No college level prerequisites are assumed. Where mathematical notions are employed, intuitive discussions are included to help the student relate to and understand the result.

The book is language independent—we present algorithms in flowcharts, pidgin algol, or decision tables. We show algorithmic methods of translating from one form to another (The text has been taught with at least the following languages: Basic, Cobol, Fortran, PL/1, Algol 60, DELISA—a lisp dialect and APL. It has been class-tested in introductory, computer science together with a variety of language primers. The objective of the text is to provide material at a depth such that it can be covered at the rate of one chapter per week in a one-term course. It is suggested that Chapters 1 through 7 be covered thoroughly, and that topics be selected from the remaining chapters according to the interests of the students and the instructor. It is also possible through more intense work in applications (Chapters 8, 9, and 11) to support a full year of instruction.

The core material is contained in the following sections: 1.1, 1.2, 1.3, 2.1, 2.2, 4.1, 4.3, 4.4, 5.1, 6.1, 6.2, 6.3, 6.4, 6.5, 7.1, 7.2, 7.3, 7.4, 7.5. The remaining material in chapters 2, 4, 5, and 6 is optional and may be covered in sequence. Chapter 3 may be covered after chapter 2 or deferred until all of the core material has been covered—at which time the student will surely have a deeper appreciation of the many concepts and techniques relevant to structuring.

Chapters 8-9 and chapters 11-15 may be covered any time after the core material has been covered. The order is also not critical. A second course in general computer science can easily cover all of the remaining material. Chapter 10 may be covered at any time after chapter 1, with additional outside readings recommended, such as Norbert Weiner's *God and Golem.*

Although the complete text is designed for a one year course, we have also taught it extensively as a one semester course using selected material. Students who complete the year course have a preliminary working knowledge of both high level language programming—including recursion and other advanced topics—as well as a knowledge of structured assembly language programming. (Kimura)

Acknowledgements

The authors are particularly indebted to students and reviewers who have provided helpful suggestions in preparing the text. The faculty and administration of the University of Delaware deserve special thanks for their support of the objectives of Foundations of Computer Science and for providing the academic means for encouraging its development. The patience and persistence of Miss Beverly Crowl in typing the text is also gratefully acknowledged. For the understanding and encouragement from our families, no amount of praise or thanks would be sufficient.

Contents

Chapter 1

Computer Science: Scientific and Historical Perspectives

In commenting about the relation between experience and its assimilation, Frederick the Great is said to have observed that he knew two mules who had been in the army for forty years but were still mules.

This story applies directly to computers and computer science. A computer scientist is one who not only works with computers, but also abstracts the essential principles of the computer and its application. In learning computer science, you will learn to use the computer, but you will also learn a discipline of thought. The methods of formulating and solving problems which are basic to computer science are also applicable to many other sciences, arts, and humanities.

We begin with an overview of the field of computer science. As a young science, it is still rapidly expanding, with constantly shifting frontiers. Still we try to delimit the scope of the science, and give an indication of how it applies to other fields.

The historical background of computer science is also sketched briefly. The history of the subject is often presented purely as a chronology of devices from mechanical to microelectronic. However, there are also general philosophic and scientific roots to the attempted mechanization of mathematics and of thought.

1.1 Overview of Computer Science

Computer science is an emerging discipline of thought and activity which may be dated from the early 1960s, although its historical roots in logic and mathematics go back to the turn of the century. The very newness of the discipline leads to self-consciousness and introspection which is not found in the more classical disciplines of science. The definition of computer science is still an active area of discourse and development from which only a few glimmers of insight have emerged so far.*

The following is a *tentative* definition of computer science:

> Computer science is a discipline concerned with the study of the principles of programming and problem solving via digital computers, and of representing and processing information.

In order to fully understand this definition, the terms "programming" and "digital computer" deserve some analysis. *Programming* is the act of organizing a complex set of related activities into correct, optimal sequences. Thus, the route selection in travel, pattern layout in sewing, and even the strategy

---

*"It is only after long experience that most men are able to define a thing in terms of its own genus, painting as painting, writing as writing."

Ezra Pound, "A, B, C of Reading," 1934, p. 71

used in courting are examples of programming. Some programming principles are:

(1) *Invariant Imbedding:* In solving a programming problem one should seek smaller, simpler programming problems imbedded within the original one.
(2) *Isolation:* Insofar as is possible, a programming task should be isolated
(3) *Generality:* The solution to the programming problems should be applicable to other similar problems.

Exercise 1.1-1

Propose a travel itinerary for road travel from New York to Los Angeles in minimum time. What principles of programming were used in arriving at your solution?

Exercise 1.1-2

Consider the Towers of Hanoi problem which requires that you move the discs in Figure 1.1 from peg No. 1 to peg No. 3, such that
(a) Only one disc may be moved at a time;
(b) No disc may rest upon any other disc of smaller size.

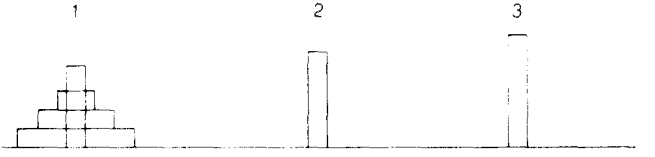

Figure 1.1: Towers of Hanoi

Discuss how you solved the problem. What principles of programming were involved? (Hint: First solve the problem of moving the two smallest discs to peg No. 2.)

Computers require organization of their activities into correct, optimal sequences. The process of organizing the actions of computers is known as *computer programming*.

A *digital computer* is an electronic device composed of interconnected discrete state elements. The interconnection of these elements permits such actions as counting, arithmetic, logic, and storage. The organization of the interconnections is of considerable scientific and engineering interest, but is of very limited concern to computer science. The relationship of a computer scientist to the organization of a digital computer is analogous to the relationship of an architect to a building. The primary concerns of the computer scientist are (1) the relationships between the major components of the

computer and (2) the relationships between the computer and its environment.

Further insight into the definition of computer science may be achieved by considering the theoretical foundations of the science. An outline of the main theoretical areas of computer science was cited by Salton* and attributed to H. Zemanek:

> (1) a *theory of programming*, with emphasis not mainly on the problem of distinguishing the computable from the noncomputable, but rather on a practical theory of algorithms concerned with the construction of economical and efficient programs.
>
> (2) a *theory of process and processor organization*, which takes into account the finite dimensions of existing memories, the availability of storage hierarchies of varying access speed and costs, and the desire for a reduction in computation and program production time;
>
> (3) a *theory of description* for processes and computational structures in terms acceptable to the processor; and
>
> (4) a *theory of computer applications* which would include all features common to most numeric and nonnumeric applications.

The mention of these theoretical areas does not suggest that they are well developed. Computer scientists are actively establishing suitable theoretical foundations of the discipline's primary concerns.

In defining computer science, some things that are not computer science should also be considered. The following are related to but don't coincide with computer science nor are included completely within computer science:

Digital Electronics
Mathematics
Logic
Mathematical Programming
Numerical Analysis

If you still feel unsatisfied after learning what is and what is not computer science, then perhaps the chapters that follow will provide that extra "feel" for what computer science is about.** To summarize this first view of computer science, we may say that computer science is now trying to develop programming techniques and significant ways to talk about programs as they relate to digital computers.

---

*G. Salton, "What is Computer Science?" *Journal of the Foundation for Computing Machinery* 19, No. 1, (January, 1972): 1-2. Copyright 1972, Association for Computing Machinery, Inc., reprinted by permission.

**Description would but make it less;
'Tis what I feel, but can't define
'Tis what I know, but can't express"

Beilby Porteus, "On Love"

1.2 Computer Science in Scientific Perspective

Computer science developed into a discipline during the 1950s and 1960s, a period of great emphasis upon technological progress. During this same period, economic and social pressures began to press the computer more and more into the lives of the ordinary people. Much of the interface between computer science and other sciences and humanities is changing very rapidly, but it does deserve some scrutiny in order to view future developments in their proper perspective.

The growth and development of computer science closely parallels that of nuclear science. From the beginning of the so-called "nuclear age" the computer has been relied upon to provide accurate and timely predictions of intensity of nuclear phenomena. From the design of nuclear weapons to the design and control of the most sophisticated nuclear reactors, the digital computer has been indispensable. The technology of computers also underwent rapid development under pressure from nuclear science. The technological activity is still evident in the name of the largest organization of computer-oriented professional people, "Association for Computing Machinery," emphasizing the mechanical or machine aspect of computing. Most computer scientists are now concerned with programming and problem solving.

Close upon the heels of the development of nuclear science came the rapid development of aerospace science in the 1960s. Many of the computer applications developed in nuclear science were readily transferred to aerospace applications. These aerospace applications included guidance and control computations and analysis of radiation effects, both requiring a high degree of numerical accuracy. Many of the reliability problems associated with nuclear science also arose in aerospace science. The most significant new role of the computer in aerospace science was in the area of communication. The computer was called upon to absorb, discriminate, filter, and even improve the features of data being transmitted over interplanetary distances. Instead of being a design engineering tool for a few scientists, the computer became a node in a communications network supporting space travel. Thus, in many ways the aerospace activity gave impetus to the development of the computer.

The rapid growth of population and need for service in the 1960s placed their own unique pressures upon the development of computer science. Businesses, governments, and institutions found that, due to the number of individuals being served, they needed the computer's speed and reliability. New problems arose, however, when it was realized that such applications required storing and manipulating vast amounts of data, most of which was nonnumerical. The technology of data storage underwent very rapid development under this pressure. New problems arose in the areas of data integrity

and security, design of interfaces between the nonscientist and the computer, and the elimination of useless data.

Exercise 1.2-1

List the ways you have interacted with a computer in a typical day. List the ways that you suspect you have interacted with a computer but are not sure.

1.3 Computer Science in Historical Perspective

A. Hardware Technology

Most of the concepts implemented in the earliest digital computers built during World War II had been developed by Charles Babbage during the early nineteenth century. However, the technological requirements of the design, precision, and maintenance of tolerances precluded their successful execution. When the technology of scientific instrument design had advanced sufficiently to allow the construction of these complex devices, they were essentially reinvented. It was then that the pioneering work of Babbage was rediscovered.

The principle operation of computers can be understood by referring to Figure 1.2. There are two essential components in Figure 1.2: a storage or memory for numbers or data, and a processing unit. The data to be operated on are introduced initially via the input/output unit into the storage unit. Subsequently, they are routed to the processing unit and back to storage (perhaps several times, depending on the complexity of processing) and finally the (refined) data are routed to the output.

Since the cost of storage increases with the speed of retrieving information, most computing systems use several types of storage. The fastest storage is *main storage* and is often mounted on the chassis of the central processor. *Bulk storage,* typically in the form of tapes or discs, is used to retain large files which would cost too much in main storage and to store data not currently being processed. Often, an initial phase of processing is the transfer of data from a bulk storage file to main storage. The final phase after processing is the return of the updated file from main storage to bulk storage.

In Babbage's early design and in the first implementations, the storage was provided in the form of electromechanical components similar to many office machines, and the sequence of operations on data was preset or externally controlled. The paths of information flow were mechanical linkages. The arithmetic operations were performed in seconds.

By the mid-1940s the much greater speed of electronic circuitry had replaced the electromechanical devices, and the first generation of digital computers to appear commercially in the early 1950s were electronic. These computers

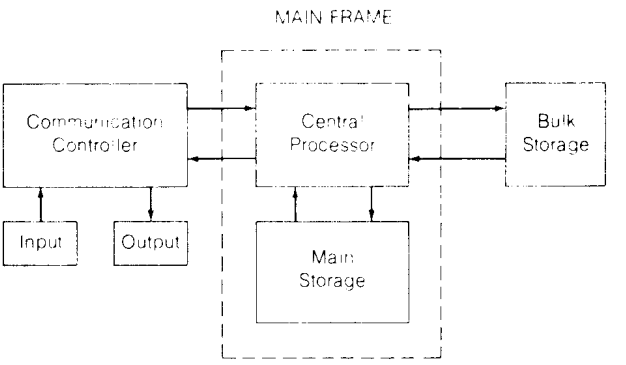

Figure 1.2: A simplified block diagram of a digital computer

used vacuum tube circuitry in the processing unit, and a variety of devices for storage. Typical arithmetic operations were measured in tens of microseconds, representing a great speed advantage over the electromechanical components. A significant innovation, introduced by von Neumann in the early electronic digital computers, was the control of the central processor activity by data stored in the memory. This *stored program* concept allows programs to be modified, as if they were data, and is used in all subsequent computers.

By the late 1950s, a new generation of computers had supplanted the early electronic digital computers. These new computers used solid state electronics in both the memory and processing units. The transistor, developed in 1948, is a miniature crystalline device functionally equivalent to a vacuum tube. It became the primary active component in the processing unit. The magnetic core memory, composed of myriads of miniature ferrite ceramic cores, became the standard memory component. Typical processing speeds for arithmetic operations were several microseconds (1 microsecond = 10^{-6} seconds).

Today's computers represent the third generation in the evolution of digital computers. Processing units are primarily composed of modules of solid state integrated circuitry. These solid state integrated circuits comprise tens to hundreds of transistors on a small crystalline base. Speed of operation of the processing circuits is measured in nanoseconds (1 nanosecond = 10^{-9} seconds). Memories are primarily magnetic core, but the cores are generally improved, smaller and faster. Solid state memories are becoming more prevalent. Typical processing speeds are of the order of one microsecond or faster for arithmetic operations.

The components we have discussed in this section are generally referred to as the *main frame of the computer.* The complete computer system will include input devices such as punched card readers, output devices such as high speed line printers, and additional storage in the form of magnetic

disks and magnetic tape units providing larger quantities of slower but more economical storage. In general, the performance of these auxiliary devices has improved correspondingly.

Finally, as the performance of components has increased, the price has decreased, and the reliability has improved. A good example of the effect of this advance in technology is that today's mini-computers sell for a great deal less than the largest computers of the early 1950s and provide superior performance. On the other hand, the large computers of today that are comparable in price to their predecessors provide significant increases in the size and performance of their storage and processing components.

The improvements in design have altered the use of computers in several ways: (1) A larger class of problems is solved by computers now; (2) A greater emphasis is placed on the use of the computer to simplify problem solving; (3) Some apparently inefficient utilizations are tolerated to make the user's job easier.

In summary, the past several decades have seen a continuing development of computer technology, with the promise of equal progress in the foreseeable future. The improvement in performance is likely to alter the user's view of the computer.

B. Mathematical Roots

We have a different view of the history of computer science when we look at the development of the mathematical concepts. There is a long history, going back several centuries, of the desire to reduce mathematics to mechanical, routine procedures. With the development of mathematical logic in the last half of the nineteenth century and early twentieth century, this goal began to look feasible. A plan to make mathematical problem-solving routine is attributed to Hilbert. The routine solution of a mathematical problem is known as the *algorithmic method*. Given any pair of numbers, x and y, we can compute their product, even though we have never performed that particular multiplication before. Likewise, given a theorem of geometry, e.g. $x^2 + y^2 = z^2$ in a right triangle where x, y are the lengths of the legs and z is the length of the hypoteneuse (Pythagorean Theorem), we would like to have a routine method of generating the proof.

Attempts to achieve the mechanization of mathematics led to some very startling results. The following results were proved in the 1930s: (1) No matter how we develop our mathematical system, there will be true facts which we will never be able to prove (2) Given a sufficiently powerful mathematical system to do arithmetic, we will be unable to automatically ascertain whether a proposed theorem is, in fact, provable.

As a result of these developments and a recognition of the limits of the algorithmic method, logicians and students of the foundations of mathematics began to describe what could be computed—even though there were no

automatic computers. The objective was to describe, or characterize, the types of problems which one could solve by a systematic (perhaps manual) computation, and what, in fact, would qualify as a systematic (or algorithmic) computation.

Due to the work of these logicians (Turing, Church, Kleene, Post, Markov) we began to develop some concept of an algorithm and a programming language (or language of algorithms). Thus, this work forms some of the theoretical basis of computer science.

C. History of Programming

As we have seen, as computers became more complex and the problems they had to solve became more difficult, the task of programming, or preparing the problem for the computer, grew. Moreover, as the computer technology developed, the number of different kinds of computers grew, making the problem of transferring programs between computers more significant.

The solution to these problems was to program in an algorithmic language and to use the computer to translate the algorithmic language, problem-oriented program to a machine language program.

Fortunately, the field of linguistics was also undergoing a revolution as a result of the same mathematical logic ideas being applied to language. The linguistic developments have made the formal description of the algorithmic languages and the translation from algorithmic to machine language easier.

The earliest algorithmic language of significance was FORTRAN I, developed in the mid-1950s. It is difficult to recreate the atmosphere of only twenty years ago when the feasibility of a machine independent algorithmic language was seriously questioned. Today, FORTRAN is extensively used for scientific computation, with a FORTRAN compiler available for almost every extant computer. (A *compiler* is a program that translates a program from an algorithmic language to a machine language.)

Sammet* discusses some 170 problem-oriented, machine-independent languages. Of these languages, approximately half are specially designed for some limited class of application, but the remaining half are general purpose with more than a dozen enjoying a wide distribution and popularity.

COBOL, developed in 1960, is a business-oriented, machine independent, algorithmic language, and probably the most widely used procedure-oriented language. ALGOL, developed at about the same time, is a language that has had wide influence, especially overseas and in academic environments.

In the mid-1960s, PL/1 was developed to combine many of the features of FORTRAN, ALGOL, and COBOL in a single language. It has not yet succeeded in replacing any of these languages significantly.

*Communications of the Association for Computing Machinery (1970).