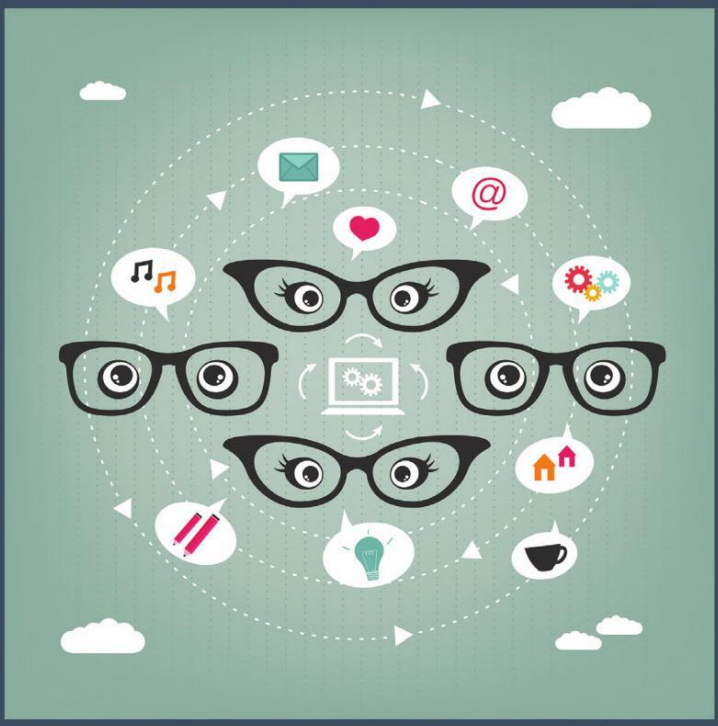
CHAPMAN & HALL/CRC
TEXTBOOKS IN COMPUTING

COMPUTATIONAL THINKING FOR THE MODERN PROBLEM SOLVER



DAVID D. RILEY
KENNY A. HUNT

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

CHAPMAN & HALL/CRC
TEXTBOOKS IN COMPUTING

COMPUTATIONAL THINKING FOR THE MODERN PROBLEM SOLVER

DAVID D. RILEY AND KENNY A. HUNT
University of Wisconsin
La Crosse, USA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20140206

International Standard Book Number-13: 978-1-4665-8779-3 (eBook - PDF)

Contents

Preface

> Computational thinking is a fundamental skill for everybody, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytic ability.
>
> —**JEANNETTE WING**[*]

Traditionally, general education courses in computer science have been rooted in some combination of four topics: (1) computer programming, (2) computer hardware, (3) societal issues of computing, and (4) computer application skills. Computational thinking is different because the focus goes beyond introductory knowledge of computing to treat computer science as an independent body of thought that is an essential part of what it means to be educated today. Thinking algorithmically is uniquely important just as is scientific investigation, artistic creativity, or proof theory in mathematics; and yet computational thinking is a distinct form of thought, separate from these other academic disciplines. The diagrammatic techniques used in software engineering analysis are effective for such efforts as strategic planning. The way that data is digitized has a profound impact on today's graphical art and music. Computer science modeling techniques are essential in many aspects of today's research in the social sciences and business. Pattern-matching techniques are useful in even the most rudimentary forms of DNA analysis. Understanding things such as how to express software requirements and the limits of computing are essential for all people who expect to live and work in a world where information is stored, accessed, and manipulated via computer software.

This book adheres to the concept of computational thinking. Since content such as this is typically taught in more advanced computer

[*] Dr. Jeannette Wing is assistant director, Computer and Information Science and Engineering Directorate, National Science Foundation and former dean of the School of Computer Science at Carnegie Mellon University.

science courses, special attention is paid to the use of effective examples and analogies. In addition, every effort is made to demonstrate the ways that these concepts are applicable in other fields of endeavor and to keep this material both accessible and relevant to noncomputer science majors.

The primary topical threads of this presentation can be grouped into foundational computer science concepts and engineering topics. The foundational computer science threads include abstraction, algorithms, logic, graph theory, social issues of software, and numeric modeling. The engineering threads include execution control, problem-solving strategies, testing, and data encoding and organizing. Rather than organize all chapters around these threads, a more logically connected approach is employed. So, for example, algorithmic thinking is integral to at least six different chapters as a part of problem solving, control structures, modeling, correctness, limits of computation, and concurrency.

It is expected that anyone teaching a computational thinking course will include some instruction in computer programming. However, there are many suitable programming languages and various depths of coverage that might be appropriate. Therefore, this book does not include computer programming instruction per se. However, the fundamental concepts of programming—variables and assignment, sequential execution, selection, repetition, control abstraction, data organization, and even concurrency—are presented. Particular care has been given to present algorithms using language-independent notation.

This approach has been taught, using early manuscript versions of this book, for several semesters to university students. Reactions have been largely positive from both the students and the several faculty involved.

Authors

David Riley has been committed to computer science education for more than 35 years. He has authored eight other computer science textbooks, along with numerous book chapters and research papers. His interest in computational thinking spans countless experiences teaching computer science majors and graduate students, as well as nonmajors, and even a year as a high school teacher. He has taught a full array of computer science courses. Jeannette Wing's seminal paper, titled "Computational Thinking," and Wing's subsequent discussions at the University of Wisconsin–La Crosse caused Riley to reconsider the priorities of a computing-related education, especially as they pertain to students outside the computer science mainstream. For the past three years he has taught several sections of computational thinking to students not intending to study any other computer science. This book is based upon these experiences.

Kenny Hunt has more than 25 years of experience in the fields of computer science and engineering. His technical expertise spans a broad array of the computational spectrum: from the design of research satellite electronics to the development of large-scale cloud-based web applications. He has authored numerous research articles and published a text on image processing. He has taught computer science and software engineering to both graduate and undergraduate students for more than 15 years and is greatly intrigued by the educational benefits of computational thinking.

CHAPTER 1

What Is Computational Thinking?

Computational Thinking—It represents a universally applicable attitude and skill set everyone, not just computer scientists, should be eager to learn and use.

—JEANNETTE WING

OBJECTIVES

- To provide a working definition for the concept of computational thinking
- To introduce the distinction between analog and digital representations of data
- To examine the origins of mechanical calculation using the abacus as an example to represent, store, and process data
- To examine key historical events that contributed to the invention of modern computing hardware and software
- To explain the stored program concept and the role it plays in software execution and the manipulation of data
- To introduce the basic components and characteristics of a modern computer
- To explain Moore's law and its impact

Is there any human invention that has changed the world more than the computer? Certainly this is a question worthy of discussion. We live in a time when not owning a computer puts a person at a disadvantage in countless ways. Apart from desktop computers, laptop computers, and tablet computers, many other of today's devices rely upon embedded

computers. Traction control, antilock brakes, computer-assisted parking, and even car repair all involve computers on board automobiles. Digital cameras are little more than a computer with a lens attached and most cell phones are really just handheld computers.

1.1 COMPUTERS, COMPUTERS EVERYWHERE

Computers impact nearly every aspect of life. Among the first occupations to rely upon computers were accounting and engineering, utilizing the speed and accuracy of computers for complex calculations. Later, writers, scholars, and journalists began to rely upon word processing for efficient ways to create and modify documents. Clearly, graphic artists and motion picture animators depend heavily upon computers. Consider the glass of milk you drank for breakfast. This milk most likely originated with genetically engineered crops fed to cows in rations determined by a computer chip around the cow's neck, while a computer-controlled robot milked the cows, and there were myriad computers involved with transporting, processing, and retailing the milk before you brought it home to your computer-controlled refrigerator.

Today, our finances are computer managed, our wars are fought increasingly by computer-controlled devices, and we frequently communicate with our friends through computer-reliant social networks. Unfortunately, even the fastest growing form of criminal activity is categorized as "computer crime."

The point is that you really don't have any choice about the limitless impact computing has on your life. The only choice is how to respond; you can choose to educate yourself about computers and learn to use them to your advantage, or you can choose the path of the luddite. (The word "luddite" was included in the English language not so long ago, specifically to label the person who is technology ignorant.)

1.2 COMPUTER, COMPUTER SCIENCE, AND COMPUTATIONAL THINKING

We use the terms *computer* or *computer system* to refer to a collection of computer hardware and software.* Computer *hardware* includes all of the physical devices that collectively constitute the item we think of as a

* Technically, it is more precise to use the term *computer system* to refer to hardware plus software, and restrict the meaning of *computer* to hardware only. However, since computer hardware is of little value without software, it is now common to use the term "computer" to mean either hardware only or hardware plus software.

desktop or a laptop computer. Such items as keyboards, LCD, computer memory, disk drives, CD and DVD drives, mice and track pads, and processors are typical parts of computer hardware.

But the computer hardware of even the most sophisticated of all computers would be of no practical value were it not for computer software. The term *software* refers to any group of computer programs. Perhaps the most important difference between a computer and other machines is the computer's ability to respond to instructions, and the instructions for performing a certain task are called a *program*. It is also acceptable to use the word *code* in place of software or program.

You have encountered numerous computer programs if you have used a computer. When you went surfing about the Internet, you were using a web browser program, such as Chrome or Internet Explorer or Firefox or Safari. Among the first things people do with a newly purchased computer is to configure the antivirus software. A computer program running on your computer might allow you to play music that was downloaded by way of a computer program running on another computer located somewhere on the Internet. Computer programs do everything from managing your bank account to formatting the pages of this book. Whenever you download any *app* to your cell phone, you have just installed a program.

Whereas most human inventions are designed to perform a specific task, computers are set apart from other machines because of the variety of tasks the computer can perform. So long as someone can create the program, the computer can perform the associated task. Often, these programs are called *applications* in recognition that the program is simply a way to apply the computer hardware to a specific purpose.

Not surprisingly, people whose career is creating programs have titles such as *programmer* or *software developer*. Since every program is designed to satisfy someone's requirements, the program is in effect solving a problem. This means that programmers are really a kind of problem solver; and given the importance of computers in our lives, computer programmers are arguably the most important of all modern problem solvers.

So how does *computer science* fit into this discussion of computer hardware and software? It turns out that study of computer science includes all issues surrounding computers from hardware to software, from the foundational theories of the technology to the end-user applications. Subfields of computer science such as computer architecture explore the way in which electrical circuits are designed, whereas software engineering examines the preferred techniques for analyzing problems, and designing

and implementing programs to solve them. Some subdisciplines of computer science, like graphics, robotics, information security, networking, and artificial intelligence, study the concepts implied by their names. All of these computer science topics, and others, play a role in this book.

The preceding discussion has been leading to the central issue of this book, namely, *computational thinking*. The best way to characterize computational thinking is as the way that computer scientists think, the manner in which they reason (Figure 1.1).

Of course it is not possible to explore everything that is known to computer science. So we have selected computer science concepts, techniques, and methods that have the widest utility to those individuals who most likely will not be computer scientists. In other words, this is written to capture how computer scientists think for the rest of us. Some of the book's topics are necessary simply to be literate in a society that is so dependent upon computers. Some of the concepts will allow you to more effectively use computers in your own field. Many of these ideas are borrowed from more advanced computer science courses. However, it is increasingly the case that computing concepts are used everywhere. Words like "multitasking," "downloading," and "flash memory" illustrate how computer science jargon has found its way into everyday speech. Discoveries in many fields would not have been possible without computers. Human genome sequencing requires the processing of thousands of genes made from billions of base pairs. Motion pictures rely on computational techniques, such as wire frame models, to create lifelike images of fictional

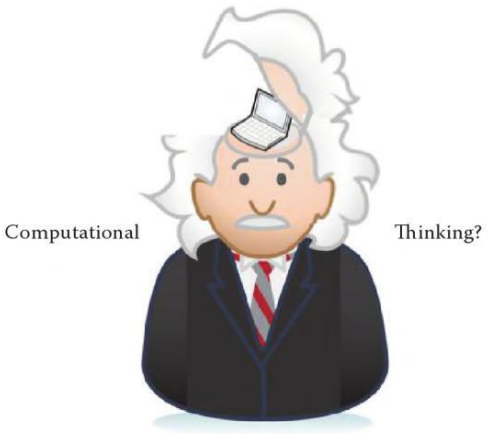


FIGURE 1.1 Computational thinking?

worlds. Modern medicine is practiced with minimal invasiveness due to robotics.

The scientific community discovered roughly a decade ago that most future scientific discovery would require computing knowledge among the researchers. As a result, new specialties, such as computational biology and computational physics, have become common in institutions of higher education.

But computational thinking is useful well beyond the scientific community. A computing subfield known as "artificial intelligence" has led to significant discoveries in psychology. Many software engineering tools used in software design have proven to be highly effective as business management tools. Computer programs have revolutionized how music is written, and architects use computer imagery to visually "walk about" buildings long before they are built. In short, computers allow us to study things that were previously too small, too large, too distant, too fast, or too complex. But as every good carpenter knows, you cannot get the most out of a tool unless you know how to use it.

1.3 FROM ABACUS TO MACHINE

We begin the history leading up to modern computers by considering calculating devices, because an important aspect of computer hardware is the ability to perform calculations. Certainly, the earliest known calculating device is the *abacus*. Although it is believed that the abacus was used in Mesopotamia centuries before, the oldest archaeological evidence of an abacus dates back to approximately the fifth century BC and the oldest known written description of an abacus is estimated to have been written in China in the thirteenth century AD.

There are variations on the basic structure of this device; we shall examine a version most commonly used in recent years and known as the "Chinese abacus" (see Figure 1.2).

The abacus consists of beads strung onto spindles. Each spindle is supported from its ends, as well as through a bar offset from its center. The number of spindles can differ from one abacus to another. The bar separates the beads into two groups. The key thing to remember while using an abacus is that every bead should be pushed as far as possible toward one end of the spindle or the other. In other words, no bead should ever be positioned to allow more than one bare space (region of exposed spindle) on each side of the bar.

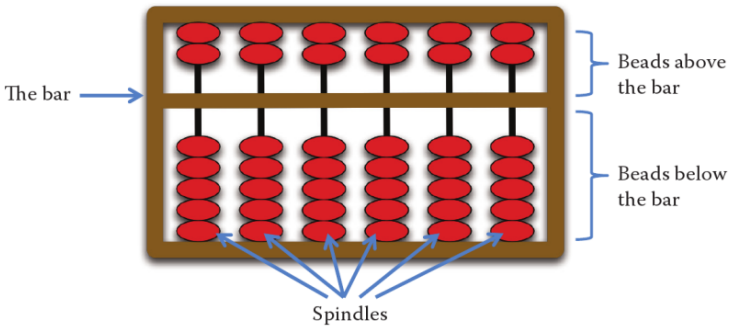


FIGURE 1.2 The Chinese abacus.

The beads have values that increase right to left, just like the value of digits in a decimal number or a Roman numeral have increasing value from right to left. The rightmost spindle of beads below the bar are called the 1s beads because each has a value of 1, while the beads of the rightmost spindle above the bar are the 5s beads. For the second spindle from the right below the bar are 10s beads and above the bar are 50s beads. The third bar from the right has 100s beads below and 500s beads above, and so forth.

Only the beads pushed as close as possible to the bar contribute to the value. This means that to make the abacus represent the value 4 you should push four 1s beads against the bar and all other beads away from the bar. Figure 1.3 illustrates both the value 4 and the value 2,639 as they could be represented on an abacus.

Different kinds of abacus may have different numbers of beads on each spindle, but the Chinese abacus always has two beads above the bar and five below. This configuration allows the abacus to represent most numbers in multiple ways. For example, Figure 1.4 shows three different ways to represent the number 10.

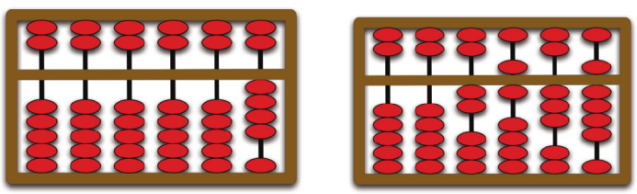


FIGURE 1.3 Two abacus configurations and their values.

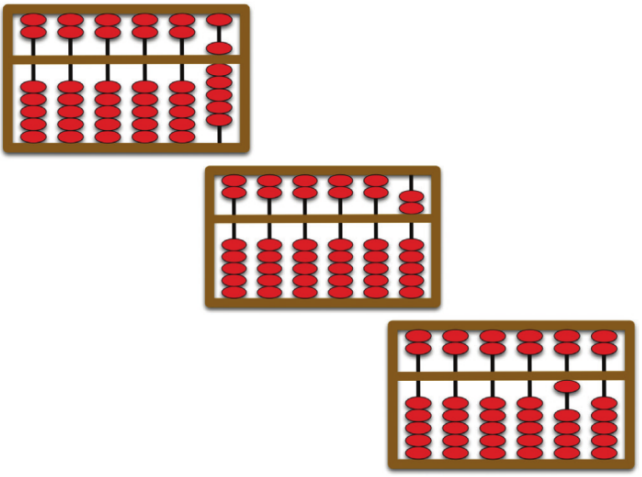


FIGURE 1.4 Three ways to represent 10 with a Chinese abacus.

Modern computers borrow four concepts from the abacus:

1. Storage
2. Representation
3. Calculation
4. User interface

For any valid bead configuration we can think of the abacus as storing the associated numeric value. So long as the beads are not moved, the abacus retains this same numeric value. A significant aspect of a modern computer is its storage. Of course, your computer can store much more than a single number, and it does not use beads, but both the abacus and your computer are definitely capable of storage.

If there is storage, then there must be something to store. The items that are stored are commonly referred to as *data*. An abacus can only store a single datum at any point in time, while your laptop can store trillions of pieces of data.

The second concept your computer borrows from the abacus is the notion of *representation*. A representation occurs anytime the data from one system is intended to model something else (the information being represented). The abacus stores (represents) an integer, using beads on a spindle to do so. The location of beads can be translated into a numeric

value—the value that is represented. A modern computer is designed to solve problems that involve real-world information. That information is represented as data within computers using various technologies, many of them electronic. The electronic signals inside your computers memory can be translated into the information that they represent. We will discuss just how computers represent information in more detail in Chapter 2.

The third property of a computer also present in an abacus is the ability to perform calculations. Truthfully, neither the abacus nor computer hardware alone can perform calculations. In the case of the abacus something (usually a human) must push the beads around. Addition and subtraction are possible by adding or removing beads next to the bar. As mentioned before, computer hardware also requires something, namely, software, in order to perform calculations. Just like humans can cause an abacus to perform arithmetic, software can cause computers to perform computations.

As a final similarity to modern computers, the abacus illustrates the first known user interface for a calculating device. The term *user interface* refers to the way that humans communicate with the machine. In the case of the abacus the user interface consists of the use of fingers and thumbs to slide beads mounted on spindles and to visually interpret the represented value by the location of beads. The user interface on your laptop computer is much more sophisticated, using a keyboard and a trackpad together with some kind of liquid crystal display (LCD). We say that you use a *graphical user interface (GUI)* because most computer interaction involves the manipulation of graphical images, such as icons, buttons, sliders, pop-up windows, and pull-down menus.

The abacus may exhibit some concepts still in use by today's computers, but no one would use the word "computer" to describe an abacus. The abacus does not have enough storage, is designed to represent only integers, is limited in the kind of calculations it can perform, and has a rather crude user interface.

The importance of improving the calculation capabilities of human inventions was evident for many centuries after the abacus. One example device of note was *Napier's bones* invented by a Scottish mathematician named *John Napier* and published in 1617. Napier's bones consist of small rectangular sticks with numbers and lines on each stick. Different sticks have numbers positioned in cleverly different ways (Figure 1.5). Arranging the sticks in different ways makes it convenient to perform multiplication, division, and even calculating square roots.

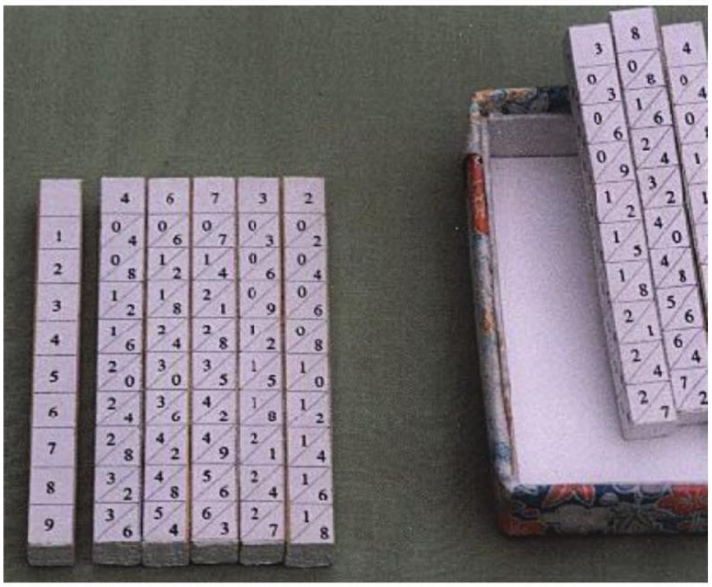


FIGURE 1.5 Napier's bones.

The next step toward computer hardware improved both the speed of calculation and user interface, while bringing human invention into the category of something that could validly be called a "machine." Actually, there were a few inventions that occurred in history during roughly the same time. These first calculating machines were invented by mathematicians and from various countries in Europe. Perhaps the two most significant of the earliest mechanical calculators were *Pascaline*, invented in 1643 by Frenchman *Blaise Pascal*, and *Leibniz' calculator* invented by the German mathematician and philosopher *Gottfried Leibniz* around 1674. Figure 1.6 and Figure 1.7 show photos of these machines.

Pascaline and the Leibniz' calculator advanced the user interface by permitting the user to turn cranks and thumb wheels. These devices also did a better job of assisting humans through the use of internal wheels, gears, and levers that accomplished addition, subtraction, multiplication, or division. These machines also demonstrate the importance of speed when performing calculations. Presumably, a knowledgeable user could perform lengthy calculations more rapidly using these machines rather than an abacus or Napier's bones. These were early devices that already illustrated man's conquest of finding machines capable of accelerating calculations.

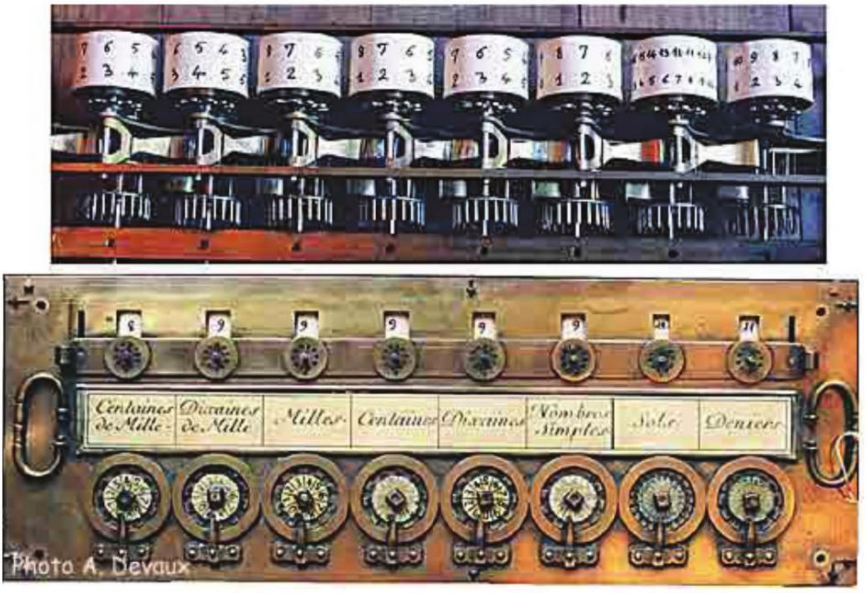


FIGURE 1.6 Pascaline.

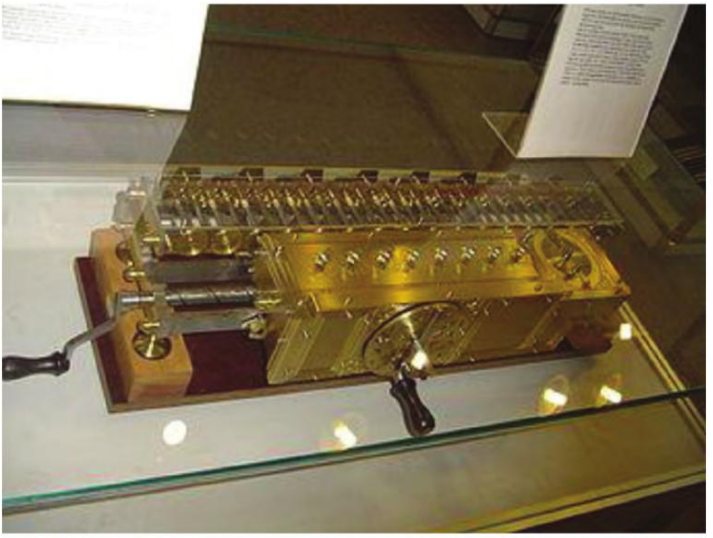


FIGURE 1.7 Leibniz' calculator.

FIGURE 1.8 Fragment of Antikythera mechanism.

Not every historical calculating device was used to perform arithmetic. The device that is the first known example of using gears for calculation is the *Antikythera mechanism* (Figure 1.8). Dated to the first century BC, this machine contained at least 30 interconnected brass gears of various dimensions. It is believed that positioning a crank on the Antikythera mechanism caused the device to accurately identify the location of the sun, moon, and planets.

The Antikythera mechanism has been called a computer by some people. However, it is probably more accurate to think of it as a special purpose calculator, somewhat related to time-keeping machines. Remarkably, fifteen to sixteen centuries would pass before the gearing technology of the Antikythera mechanism would reappear in the watch-making industry and early calculators, such as Pascaline.

1.4 THE FIRST SOFTWARE

None of the devices described in Section 1.3 were truly *programmable*. Yes, it is possible to rearrange beads, relocate bones, or turn wheels and cranks, but these are merely ways to configure devices to perform calculations. In order to perform a different calculation any prior configuration is lost. A truly programmable device is one in which the program is divorced from the hardware so that it can be stored for reuse at a different time. In other words the program "instructs" the device in how to perform, and different programs produce different results.

The first known programmable machine is not a calculator; it is a loom for weaving cloth. Around 1805, a French inventor named *Joseph-Marie Jacquard* built the first known programmable machine. The *Jacquard loom* (Figure 1.9) was similar to other looms of the day except that it used a loop of stiff paper cards as a program. The cards had holes punched in them. Changing the number and placement of holes in these cards would cause the loom to weave a different pattern. The loom was built so that the loop and cards could be removed and replaced by a different loop of cards; thereby programming the loom to weave different patterns. This kind of punch card program is still used on textile looms today.

Although punched cards might represent programmability, weaving on a loom is quite different from computer-like calculations. The first example of what might be termed "computer software" (or at least calculator software) did not occur until approximately 1843. This important event in history came from an English mathematician and inventor named

FIGURE 1.9 Model of a Jacquard loom.

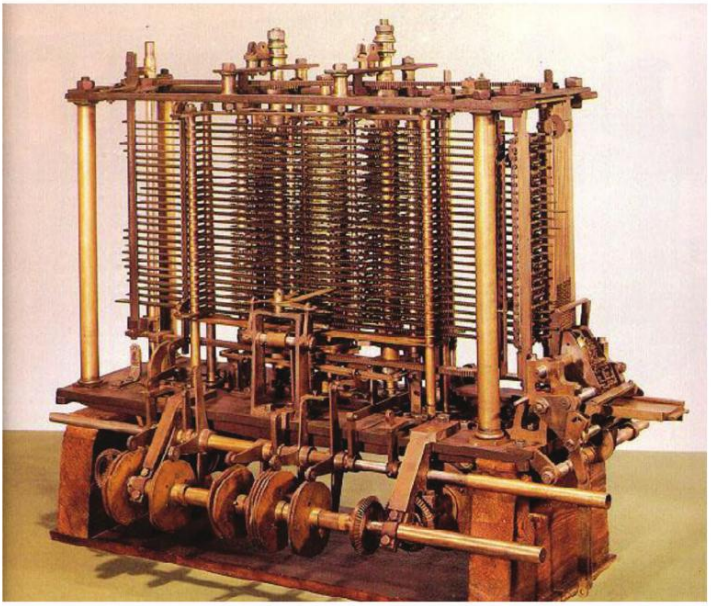


FIGURE 1.10 A piece of the Analytical Engine.

Charles Babbage. Babbage had already built a mechanical calculator capable of more advanced logarithmic and trigonometric calculations, but he did not add the notion of programmability until the design of his second, and more significant, invention—the *Analytical Engine* (Figure 1.10).

The Analytical Engine adopted the concept of *punched cards* to store and input a program into the hardware. But programs for the Analytical Engine were capable of performing a sequence of mathematical operations in the same way that modern computers can perform complex mathematical operations as directed by a proper computer program. Sadly, because of the complexity of the device, the manufacturing capabilities of the day made it impossible to construct a complete Analytical Engine during Babbage's lifetime.

An interesting side note in history often told about the Analytical Engine involves a woman named *Ada Lovelace* (Figure 1.11). The Countess Lovelace was the daughter of the famous poet, Lord Byron. She was quite interested in the work of Charles Babbage and is known to have written programs for the Analytical Engine. Some people have called Ada Lovelace the first programmer, but this cannot be confirmed and is most likely not true, since several individuals (Babbage included) wrote programs at about the same time. Nonetheless she is clearly among the first programmers.

FIGURE 1.11 Charles Babbage and Ada Lovelace were among the first computer programmers.

1.5 WHAT MAKES IT A MODERN COMPUTER?

One widely accepted definition of *modern computer* requires three properties of this calculating device:

1. It must be *electronic* and not exclusively mechanical.
2. It must be *digital* and not analog.
3. It must employ the *stored program concept*.

As it happens, even the Analytical Engine invented by Babbage fails to satisfy every one of these three requirements.

To find the first invention that is believed to satisfy at least one of these three properties, we skip to the 1890s. The United States has a long history of taking census every ten years. In 1880 the census was tabulated, like every decade prior, by hand. This process of counting citizenry and categorizing them by geographic region was becoming difficult because of rapid population growth. In fact the 1880 census was barely completed before 1890 when the next census was to begin.

A man named *Herman Hollerith* invented a calculating device built specifically for tabulating the US census. Hollerith's machine completed the 1890 census in less than one year. More important for computing, Hollerith's machine ran on electricity. The Hollerith tabulating machine can fairly be labeled as the first calculating (i.e., computer-like) hardware that satisfies any of the properties that distinguish a modern computer.

Hollerith later founded a Tabulating Machine Company to build these devices, and his company merged to form IBM Corporation in 1924; IBM remains today as one of the world's largest manufacturers of computers. Hollerith's tabulating machine also provides convincing evidence of the future capacity of computers to assist in solving human problems.

Before revealing the candidates for the first modern computer, there are two of the preceding properties of a modern computer that have yet to be mentioned. The first issue is that a modern computer must be digital. Prior to the 1930s, machines that stored data typically did so as represented using mechanical gears or electrical signals. Gears can generally be rotated to an infinite number of different angles. Similarly, electrical signals are infinitely variable in terms of voltage, amperage, capacitance, and inductance. This kind of continuous change is called *analog*. For example, an analog wristwatch often has a sweep second hand and can position the minute hand at an infinite number of positions around the dial.

A *digital* system, unlike analog systems, is one in which there are not an infinite number of possibilities and change is not continuous. Instead, digital systems restrict values to be one of a few choices. For example, hours and minutes on a digital watch are displayed as numbers. It is not possible for the minute number to display anything between 9:30 and 9:31. Most of our automobile speedometers are analog with a needle that rotates gradually as the car accelerates. However, a few cars have digital speedometers that display the current speed as a single number in either whole miles or meters per hour.

An explanation of the stored program concept requires a brief look at the major units of hardware in a modern computer. Figure 1.12 diagrams a simple desktop-style computer with three components: a keyboard, a display, and a system unit. These three components can be used to illustrate

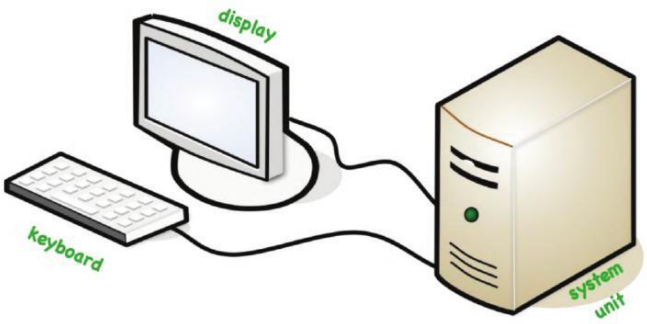


FIGURE 1.12 The basic parts of a simple desktop computer.

Clearly, ENIAC satisfies the earlier criteria for a modern computer. It was a calculating device that ran on electricity and was digital. The original version of ENIAC did not truly follow the stored program concept, but this capability was later incorporated. By today's standards ENIAC was enormous, physically filling an entire room. Its circuitry relied on 19,000 vacuum tubes (see Figure 1.14) and 1,000 relays. Vacuum tubes can be used as memory devices, but they are large (each roughly the size of a human thumb) and unreliable relative to today's computer memory. Relays are a form of electrical switch that are mechanical and also large (about the size of half a cell phone).

Despite the patent, ENIAC is not considered the first modern computer. In fact the US Patent Office invalidated the 1947 patent in 1973. The primary reason for this invalidation was the discovery of some earlier work that was not fully patented.

In 1937–1938 two physicists—*John Atanasoff* and *Chuck Berry*—at Iowa State University built a machine they called the *ABC Computer* (see Figure 1.15). During the patent dispute, it was discovered that the *Des Moines Register* had printed an article regarding the ABC Computer in 1941. It was also claimed that Atanasoff discussed his design with Mauchly in 1940 and had visited a US Patent Office that same year.

Unfortunately, the ABC Computer may not truly qualify as the first modern computer, because it failed to use the stored program concept, nor was it truly programmable for general purposes, as it was only designed to

FIGURE 1.14 Vacuum tubes.

of Manchester, England, and the first computer to use the stored program concept. However, it was never intended to be a practical computer but rather part of a test bed for other hardware. By September of the same year, ENIAC had been modified to use stored programs, making it a contender for first modern computer.

Regardless of which invention should be considered to be most significant, what is clear is that during the late 1930s and throughout the 1940s there was a flurry of research taking place around the world to create early computing devices.

Within a few years computer scientists had grown weary of the tedious activity of using machine instructions, which led to the invention of *high-level programming languages*. A high-level language is one that relies upon instructions that are much more English-like instead of the cryptic numeric form of most machine instructions. The revolution leading to computer systems like today's changed to more of an evolutionary history by the mid-1950s, except for one major change to the hardware.

1.7 MOORE'S LAW

No discussion of today's computer hardware would be complete without the inclusion of one more discovery. In the 1950s and 1960s several physicists, most notably *Jack Kilby* and *Robert Noyce*, were working on a technology that would soon replace the use of vacuum tubes and relays with smaller, faster, and far more reliable electronics.

The idea was to use silicon wafers that are manufactured in such a way that thousands, and later trillions, of electronic switches, known as "transistors," can be combined onto a single chip. Such devices are referred to as *integrated circuits* and the technology that permits silicon to function in this way is called *semiconductor technology*.

Figure 1.16 is a photograph of an integrated circuit. The blackened square region in the center is the silicon wafer with wires (the lines) connecting it to the metal legs on the outside of the device. These legs typically plug into a socket for connection to the remainder of the computer circuitry. The entire package is commonly referred to as a *chip*.

Robert Noyce was awarded the Nobel Prize in Physics for his work in the creation of semiconductors. Together with *Gordon Moore*, he founded Intel Corporation—the largest manufacturer of computer processors in the world.

Integrated circuits make it possible for us to carry computers in a briefcase or a pocket that are millions of times faster than the room-sized

TERMINOLOGY

abacus

ABC Computer

analog

Analytical Engine

Antikythera device

app

application

Attanasoff, John

Babbage, Charles

Berry, Chuck

calculation

chip

code

computational thinking

computer

computer science

computer system

data

differential analyzer

digital

Ekert, Peter

ENIAC

electronic (computer)

exponential growth

graphical user interface (GUI)

hardware

high-level programming language

Hollerith, Herman

I/O

input

integrated circuit

Jacquard, Joseph-Marie

Jacquard loom

Kilby, Jack

Leibniz, Gottfried

Lovelace, Ada

machine instruction

modern computer

Moore, Gordon

Moore's law

Napier, John

Napier's bones

Noyce, Robert

output

Manchester Small-Scale Experimental Machine (SSEM)

Mauchly, John

memory
Pascal, Blaise
Pascaline
processor
program
programmable
programmer
punched cards
representation
semiconductor technology
software
software developer
stored program concept
storage
user interface
Z4 Computer
Zuse, Konrad

EXERCISES

1. What are the three qualities required of a calculating device in order for it to qualify as a modern computer?
2. What is the difference between computer hardware and computer software?
3. Describe the significance of each of the following inventions, as it eventually led to the creation of the first modern computer.
   a. The abacus
   b. The Analytical Engine
   c. Jacquard loom
   d. Hollerith's machine for tabulating the US census
4. Digital cameras are one kind of modern computer. In this sense, answer the following questions.
   a. How does the user supply *input* to a digital camera?
   b. What would you consider to be the camera's *output* device(s)?
   c. What is the purpose of the camera's memory?