Computer science, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information. The discipline of computer science includes the study of algorithms and data structures, computer and network design, modeling data and information processes, and artificial intelligence. Computer science draws some of its foundations from mathematics and engineering and therefore incorporates techniques from areas such as queueing theory, probability and statistics, and electronic circuit design. Computer science also makes heavy use of hypothesis testing and experimentation during the conceptualization, design, measurement, and refinement of new algorithms, information structures, and computer architectures.

Computer science is considered as part of a family of five separate yet interrelated disciplines: computer engineering, computer science, information systems, information technology, and software engineering. This family has come to be known collectively as the discipline of computing. These five disciplines are interrelated in the sense that computing is their object of study, but they are separate since each has its own research perspective and curricular focus. (Since 1991 the Association for Computing Machinery [ACM], the IEEE Computer Society [IEEE-CS], and the Association for Information Systems [AIS] have collaborated to develop and update the taxonomy of these five interrelated disciplines and the guidelines that educational institutions worldwide use for their undergraduate, graduate, and research programs.)

The major subfields of computer science include the traditional study of computer architecture, programming languages, and software development. However, they also include computational science (the use of algorithmic techniques for modeling scientific data), graphics and visualization, human-computer interaction, databases and information systems, networks, and the social and professional issues that are unique to the practice of computer science. As may be evident, some of these subfields overlap in their activities with other modern fields, such as bioinformatics and computational chemistry. These overlaps are the consequence of a tendency among computer scientists to recognize and act upon their field's many interdisciplinary connections.

Development of computer science Computer science emerged as an independent discipline in the early 1960s, although the electronic digital computer that is the object of its study was invented some two decades earlier. The roots of computer science lie primarily in the related fields of mathematics, electrical engineering, physics, and management information systems.

computer chip. computer. Hand holding computer chip. Central processing unit (CPU). history and society, science and technology, microchip, microprocessor motherboard computer Circuit Board Britannica Quiz Computers and Technology Quiz Mathematics is the source of two key concepts in the development of the computer—the idea that all information can be represented as sequences of zeros and ones and the abstract notion of a "stored program." In the binary number system, numbers are represented by a sequence of the binary digits 0 and 1 in the same way that numbers in the familiar decimal system are represented using the digits 0 through 9. The relative ease with which two states (e.g., high and

low voltage) can be realized in electrical and electronic devices led naturally to the binary digit, or bit, becoming the basic unit of data storage and transmission in a computer system.

Electrical engineering provides the basics of circuit design—namely, the idea that electrical impulses input to a circuit can be combined using Boolean algebra to produce arbitrary outputs. (The Boolean algebra developed in the 19th century supplied a formalism for designing a circuit with binary input values of zeros and ones [false or true, respectively, in the terminology of logic] to yield any desired combination of zeros and ones as output.) The invention of the transistor and the miniaturization of circuits, along with the invention of electronic, magnetic, and optical media for the storage and transmission of information, resulted from advances in electrical engineering and physics.

Get a Britannica Premium subscription and gain access to exclusive content. Management information systems, originally called data processing systems, provided early ideas from which various computer science concepts such as sorting, searching, databases, information retrieval, and graphical user interfaces evolved. Large corporations housed computers that stored information that was central to the activities of running a business—payroll, accounting, inventory management, production control, shipping, and receiving.

Alan Turing Alan Turing Theoretical work on computability, which began in the 1930s, provided the needed extension of these advances to the design of whole machines; a milestone was the 1936 specification of the Turing machine (a theoretical computational model that carries out instructions represented as a series of zeros and ones) by the British mathematician Alan Turing and his proof of the model's computational power. Another breakthrough was the concept of the stored-program computer, usually credited to Hungarian American mathematician John von Neumann. These are the origins of the computer science field that later became known as architecture and organization.

In the 1950s, most computer users worked either in scientific research labs or in large corporations. The former group used computers to help them make complex mathematical calculations (e.g., missile trajectories), while the latter group used computers to manage large amounts of corporate data (e.g., payrolls and inventories). Both groups quickly learned that writing programs in the machine language of zeros and ones was not practical or reliable. This discovery led to the development of assembly language in the early 1950s, which allows programmers to use symbols for instructions (e.g., ADD for addition) and variables (e.g., X). Another program, known as an assembler, translated these symbolic programs into an equivalent binary program whose steps the computer could carry out, or "execute."

Other system software elements known as linking loaders were developed to combine pieces of assembled code and load them into the computer's memory, where they could be executed. The concept of linking separate pieces of code was important, since it allowed "libraries" of programs for carrying out common tasks to be reused. This was a first step in the development of the computer science field called software engineering.

Later in the 1950s, assembly language was found to be so cumbersome that

the development of high-level languages (closer to natural languages) began to support easier, faster programming. FORTRAN emerged as the main high-level language for scientific programming, while COBOL became the main language for business programming. These languages carried with them the need for different software, called compilers, that translate high-level language programs into machine code. As programming languages became more powerful and abstract, building compilers that create high-quality machine code and that are efficient in terms of execution speed and storage consumption became a challenging computer science problem. The design and implementation of high-level languages is at the heart of the computer science field called programming languages.

Increasing use of computers in the early 1960s provided the impetus for the development of the first operating systems, which consisted of system-resident software that automatically handled input and output and the execution of programs called "jobs." The demand for better computational techniques led to a resurgence of interest in numerical methods and their analysis, an activity that expanded so widely that it became known as computational science.

The 1970s and '80s saw the emergence of powerful computer graphics devices, both for scientific modeling and other visual activities. (Computerized graphical devices were introduced in the early 1950s with the display of crude images on paper plots and cathode-ray tube [CRT] screens.) Expensive hardware and the limited availability of software kept the field from growing until the early 1980s, when the computer memory required for bitmap graphics (in which an image is made up of small rectangular pixels) became more affordable. Bitmap technology, together with high-resolution display screens and the development of graphics standards that make software less machine-dependent, has led to the explosive growth of the field. Support for all these activities evolved into the field of computer science known as graphics and visual computing.

graphical user interface graphical user interface Closely related to this field is the design and analysis of systems that interact directly with users who are carrying out various computational tasks. These systems came into wide use during the 1980s and '90s, when line-edited interactions with users were replaced by graphical user interfaces (GUIs). GUI design, which was pioneered by Xerox and was later picked up by Apple (Macintosh) and finally by Microsoft (Windows), is important because it constitutes what people see and do when they interact with a computing device. The design of appropriate user interfaces for all types of users has evolved into the computer science field known as human-computer interaction (HCI).

The field of computer architecture and organization has also evolved dramatically since the first stored-program computers were developed in the 1950s. So called time-sharing systems emerged in the 1960s to allow several users to run programs at the same time from different terminals that were hard-wired to the computer. The 1970s saw the development of the first wide-area computer networks (WANs) and protocols for transferring information at high speeds between computers separated by large distances. As these activities evolved, they coalesced into the computer science field called networking and communications. A major accomplishment of this field was the development of the Internet.

The idea that instructions, as well as data, could be stored in a computer's memory was critical to fundamental discoveries about the theoretical behaviour of algorithms. That is, questions such as, "What can/cannot be computed?" have been formally addressed using these abstract ideas. These discoveries were the origin of the computer science field known as algorithms and complexity. A key part of this field is the study and application of data structures that are appropriate to different applications. Data structures, along with the development of optimal algorithms for inserting, deleting, and locating data in such structures, are a major concern of computer scientists because they are so heavily used in computer software, most notably in compilers, operating systems, file systems, and search engines.

In the 1960s the invention of magnetic disk storage provided rapid access to data located at an arbitrary place on the disk. This invention led not only to more cleverly designed file systems but also to the development of database and information retrieval systems, which later became essential for storing, retrieving, and transmitting large amounts and wide varieties of data across the Internet. This field of computer science is known as information management.

Another long-term goal of computer science research is the creation of computing machines and robotic devices that can carry out tasks that are typically thought of as requiring human intelligence. Such tasks include moving, seeing, hearing, speaking, understanding natural language, thinking, and even exhibiting human emotions. The computer science field of intelligent systems, originally known as artificial intelligence (AI), actually predates the first electronic computers in the 1940s, although the term artificial intelligence was not coined until 1956.

Three developments in computing in the early part of the 21st century—mobile computing, client-server computing, and computer hacking—contributed to the emergence of three new fields in computer science: platform-based development, parallel and distributed computing, and security and information assurance. Platform-based development is the study of the special needs of mobile devices, their operating systems, and their applications. Parallel and distributed computing concerns the development of architectures and programming languages that support the development of algorithms whose components can run simultaneously and asynchronously (rather than sequentially), in order to make better use of time and space. Security and information assurance deals with the design of computing systems and software that protects the integrity and security of data, as well as the privacy of individuals who are characterized by that data.

Finally, a particular concern of computer science throughout its history is the unique societal impact that accompanies computer science research and technological advancements. With the emergence of the Internet in the 1980s, for example, software developers needed to address important issues related to information security, personal privacy, and system reliability. In addition, the question of whether computer software constitutes intellectual property and the related question "Who owns it?" gave rise to a whole new legal area of licensing and licensing standards that applied to software and related artifacts. These concerns and others form the basis of social and professional issues of computer science, and they appear in almost all the other fields identified above.

So, to summarize, the discipline of computer science has evolved into the following 15 distinct fields:

Algorithms and complexity Architecture and organization Computational science Graphics and visual computing Human-computer interaction Information management Intelligent systems Networking and communication Operating systems Parallel and distributed computing Platform-based development Programming languages Security and information assurance Software engineering Social and professional issues Computer science continues to have strong mathematical and engineering roots. Computer science bachelor's, master's, and doctoral degree programs are routinely offered by postsecondary academic institutions, and these programs require students to complete appropriate mathematics and engineering courses, depending on their area of focus. For example, all undergraduate computer science majors must study discrete mathematics (logic, combinatorics, and elementary graph theory). Many programs also require students to complete courses in calculus, statistics, numerical analysis, physics, and principles of engineering early in their studies.

Algorithms and complexity An algorithm is a specific procedure for solving a well-defined computational problem. The development and analysis of algorithms is fundamental to all aspects of computer science: artificial intelligence, databases, graphics, networking, operating systems, security, and so on. Algorithm development is more than just programming. It requires an understanding of the alternatives available for solving a computational problem, including the hardware, networking, programming language, and performance constraints that accompany any particular solution. It also requires understanding what it means for an algorithm to be "correct" in the sense that it fully and efficiently solves the problem at hand.

An accompanying notion is the design of a particular data structure that enables an algorithm to run efficiently. The importance of data structures stems from the fact that the main memory of a computer (where the data is stored) is linear, consisting of a sequence of memory cells that are serially numbered 0, 1, 2,.... Thus, the simplest data structure is a linear array, in which adjacent elements are numbered with consecutive integer "indexes" and an element's value is accessed by its unique index. An array can be used, for example, to store a list of names, and efficient methods are needed to efficiently search for and retrieve a particular name from the array. For example, sorting the list into alphabetical order permits a so-called binary search technique to be used, in which the remainder of the list to be searched at each step is cut in half. This search technique is similar to searching a telephone book for a particular name. Knowing that the book is in alphabetical order allows one to turn quickly to a page that is close to the page containing the desired name. Many algorithms have been developed for sorting and searching lists of data efficiently.

Although data items are stored consecutively in memory, they may be linked together by pointers (essentially, memory addresses stored with an item to indicate where the next item or items in the structure are found) so that the data can be organized in ways similar to those in which they will be accessed. The simplest such structure is called the linked list, in which noncontiguously stored items may be accessed in a pre-specified order by following the pointers

from one item in the list to the next. The list may be circular, with the last item pointing to the first, or each element may have pointers in both directions to form a doubly linked list. Algorithms have been developed for efficiently manipulating such lists by searching for, inserting, and removing items.

Pointers also provide the ability to implement more complex data structures. A graph, for example, is a set of nodes (items) and links (known as edges) that connect pairs of items. Such a graph might represent a set of cities and the highways joining them, the layout of circuit elements and connecting wires on a memory chip, or the configuration of persons interacting via a social network. Typical graph algorithms include graph traversal strategies, such as how to follow the links from node to node (perhaps searching for a node with a particular property) in a way that each node is visited only once. A related problem is the determination of the shortest path between two given nodes on an arbitrary graph. (See graph theory.) A problem of practical interest in network algorithms, for instance, is to determine how many "broken" links can be tolerated before communications begin to fail. Similarly, in very-large-scale integration (VLSI) chip design it is important to know whether the graph representing a circuit is planar, that is, whether it can be drawn in two dimensions without any links crossing (wires touching).

The (computational) complexity of an algorithm is a measure of the amount of computing resources (time and space) that a particular algorithm consumes when it runs. Computer scientists use mathematical measures of complexity that allow them to predict, before writing the code, how fast an algorithm will run and how much memory it will require. Such predictions are important guides for programmers implementing and selecting algorithms for real-world applications.

Computational complexity is a continuum, in that some algorithms require linear time (that is, the time required increases directly with the number of items or nodes in the list, graph, or network being processed), whereas others require quadratic or even exponential time to complete (that is, the time required increases with the number of items squared or with the exponential of that number). At the far end of this continuum lie the murky seas of intractable problems—those whose solutions cannot be efficiently implemented. For these problems, computer scientists seek to find heuristic algorithms that can almost solve the problem and run in a reasonable amount of time.

Further away still are those algorithmic problems that can be stated but are not solvable; that is, one can prove that no program can be written to solve the problem. A classic example of an unsolvable algorithmic problem is the halting problem, which states that no program can be written that can predict whether or not any other program halts after a finite number of steps. The unsolvability of the halting problem has immediate practical bearing on software development. For instance, it would be frivolous to try to develop a software tool that predicts whether another program being developed has an infinite loop in it (although having such a tool would be immensely beneficial).

Architecture and organization Computer architecture deals with the design of computers, data storage devices, and networking components that store and run programs, transmit data, and drive interactions between computers, across

networks, and with users. Computer architects use parallelism and various strategies for memory organization to design computing systems with very high performance. Computer architecture requires strong communication between computer scientists and computer engineers, since they both focus fundamentally on hardware design.

At its most fundamental level, a computer consists of a control unit, an arithmetic logic unit (ALU), a memory unit, and input/output (I/O) controllers. The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data. The control unit fetches data and instructions from memory and uses operations of the ALU to carry out those instructions using that data. (The control unit and ALU together are referred to as the central processing unit [CPU].) When an input or output instruction is encountered, the control unit transfers the data between the memory and the designated I/O controller. The operational speed of the CPU primarily determines the speed of the computer as a whole. All of these components—the control unit, the ALU, the memory, and the I/O controllers—are realized with transistor circuits.

Computers also have another level of memory called a cache, a small, extremely fast (compared with the main memory, or random access memory [RAM]) unit that can be used to store information that is urgently or frequently needed. Current research includes cache design and algorithms that can predict what data is likely to be needed next and preload it into the cache for improved performance.

I/O controllers connect the computer to specific input devices (such as keyboards and touch screen displays) for feeding information to the memory, and output devices (such as printers and displays) for transmitting information from the memory to users. Additional I/O controllers connect the computer to a network via ports that provide the conduit through which data flows when the computer is connected to the Internet.

computer chip. computer. Hand holding computer chip. Central processing unit (CPU). history and society, science and technology, microchip, microprocessor motherboard computer Circuit Board Britannica Quiz Computers and Technology Quiz USB flash drive inserted in a laptop USB flash drive inserted in a laptop Linked to the I/O controllers are secondary storage devices, such as a disk drive, that are slower and have a larger capacity than main or cache memory. Disk drives are used for maintaining permanent data. They can be either permanently or temporarily attached to the computer in the form of a compact disc (CD), a digital video disc (DVD), or a memory stick (also called a flash drive).

The operation of a computer, once a program and some data have been loaded into RAM, takes place as follows. The first instruction is transferred from RAM into the control unit and interpreted by the hardware circuitry. For instance, suppose that the instruction is a string of bits that is the code for LOAD 10. This instruction loads the contents of memory location 10 into the ALU. The next instruction, say ADD 15, is fetched. The control unit then loads the contents of memory location 15 into the ALU and adds it to the number already there. Finally, the instruction STORE 20 would store that sum into

location 20. At this level, the operation of a computer is not much different from that of a pocket calculator.

In general, programs are not just lengthy sequences of LOAD, STORE, and arithmetic operations. Most importantly, computer languages include conditional instructions—essentially, rules that say, "If memory location n satisfies condition a, do instruction number x next, otherwise do instruction y." This allows the course of a program to be determined by the results of previous operations—a critically important ability.

Finally, programs typically contain sequences of instructions that are repeated a number of times until a predetermined condition becomes true. Such a sequence is called a loop. For example, a loop would be needed to compute the sum of the first n integers, where n is a value stored in a separate memory location. Computer architectures that can execute sequences of instructions, conditional instructions, and loops are called "Turing complete," which means that they can carry out the execution of any algorithm that can be defined. Turing completeness is a fundamental and essential characteristic of any computer organization.

Logic design is the area of computer science that deals with the design of electronic circuits using the fundamental principles and properties of logic (see Boolean algebra) to carry out the operations of the control unit, the ALU, the I/O controllers, and other hardware. Each logical function (AND, OR, and NOT) is realized by a particular type of device called a gate. For example, the addition circuit of the ALU has inputs corresponding to all the bits of the two numbers to be added and outputs corresponding to the bits of the sum. The arrangement of wires and gates that link inputs to outputs is determined by the mathematical definition of addition. The design of the control unit provides the circuits that interpret instructions. Due to the need for efficiency, logic design must also optimize the circuitry to function with maximum speed and has a minimum number of gates and circuits.

An important area related to architecture is the design of microprocessors, which are complete CPUs—control unit, ALU, and memory—on a single integrated circuit chip. Additional memory and I/O control circuitry are linked to this chip to form a complete computer. These thumbnail-sized devices contain millions of transistors that implement the processing and memory units of modern computers.

VLSI microprocessor design occurs in a number of stages, which include creating the initial functional or behavioral specification, encoding this specification into a hardware description language, and breaking down the design into modules and generating sizes and shapes for the eventual chip components. It also involves chip planning, which includes building a "floor plan" to indicate where on the chip each component should be placed and connected to other components. Computer scientists are also involved in creating the computer-aided design (CAD) tools that support engineers in the various stages of chip design and in developing the necessary theoretical results, such as how to efficiently design a floor plan with near-minimal area that satisfies the given constraints.

Moore's law Moore's law Advances in integrated circuit technology have been

incredible. For example, in 1971 the first microprocessor chip (Intel Corporation's 4004) had only 2,300 transistors, in 1993 Intel's Pentium chip had more than 3 million transistors, and by 2000 the number of transistors on such a chip was about 50 million. The Power7 chip introduced in 2010 by IBM contained approximately 1 billion transistors. The phenomenon of the number of transistors in an integrated circuit doubling about every two years is widely known as Moore's law.

Fault tolerance is the ability of a computer to continue operation when one or more of its components fails. To ensure fault tolerance, key components are often replicated so that the backup component can take over if needed. Such applications as aircraft control and manufacturing process control run on systems with backup processors ready to take over if the main processor fails, and the backup systems often run in parallel so the transition is smooth. If the systems are critical in that their failure would be potentially disastrous (as in aircraft control), incompatible outcomes collected from replicated processes running in parallel on separate machines are resolved by a voting mechanism. Computer scientists are involved in the analysis of such replicated systems, providing theoretical approaches to estimating the reliability achieved by a given configuration and processor parameters, such as average time between failures and average time required to repair the processor. Fault tolerance is also a desirable feature in distributed systems and networks. For example, an advantage of a distributed database is that data replicated on different network hosts can provide a natural backup mechanism when one host fails.

Computational science Computational science applies computer simulation, scientific visualization, mathematical modeling, algorithms, data structures, networking, database design, symbolic computation, and high-performance computing to help advance the goals of various disciplines. These disciplines include biology, chemistry, fluid dynamics, archaeology, finance, sociology, and forensics. Computational science has evolved rapidly, especially because of the dramatic growth in the volume of data transmitted from scientific instruments. This phenomenon has been called the "big data" problem.

The mathematical methods needed for computational science require the transformation of equations and functions from the continuous to the discrete. For example, the computer integration of a function over an interval is accomplished not by applying integral calculus but rather by approximating the area under the function graph as a sum of the areas obtained from evaluating the function at discrete points. Similarly, the solution of a differential equation is obtained as a sequence of discrete points determined by approximating the true solution curve by a sequence of tangential line segments. When discretized in this way, many problems can be recast as an equation involving a matrix (a rectangular array of numbers) solvable using linear algebra. Numerical analysis is the study of such computational methods. Several factors must be considered when applying numerical methods: (1) the conditions under which the method yields a solution, (2) the accuracy of the solution, (3) whether the solution process is stable (i.e., does not exhibit error growth), and (4) the computational complexity (in the sense described above) of obtaining a solution of the desired accuracy.

The requirements of big-data scientific problems, including the solution of ever larger systems of equations, engage the use of large and powerful arrays of processors (called multiprocessors or supercomputers) that allow many calculations to proceed in parallel by assigning them to separate processing elements. These activities have sparked much interest in parallel computer architecture and algorithms that can be carried out efficiently on such machines.

Graphics and visual computing is the field that deals with the display and control of images on a computer screen. This field encompasses the efficient implementation of four interrelated computational tasks: rendering, modeling, animation, and visualization. Graphics techniques incorporate principles of linear algebra, numerical integration, computational geometry, special-purpose hardware, file formats, and graphical user interfaces (GUIs) to accomplish these complex tasks.

brain cancer; magnetic resonance imaging (MRI) brain cancer; magnetic resonance imaging (MRI) Applications of graphics include CAD, fine arts, medical imaging, scientific data visualization, and video games. CAD systems allow the computer to be used for designing objects ranging from automobile parts to bridges to computer chips by providing an interactive drawing tool and an engineering interface to simulation and analysis tools. Fine arts applications allow artists to use the computer screen as a medium to create images, cinematographic special effects, animated cartoons, and television commercials. Medical imaging applications involve the visualization of data obtained from technologies such as X-rays and magnetic resonance imaging (MRIs) to assist doctors in diagnosing medical conditions. Scientific visualization uses massive amounts of data to define simulations of scientific phenomena, such as ocean modeling, to produce pictures that provide more insight into the phenomena than would tables of numbers. Graphics also provide realistic visualizations for video gaming, flight simulation, and other representations of reality or fantasy. The term virtual reality has been coined to refer to any interaction with a computer-simulated virtual world.