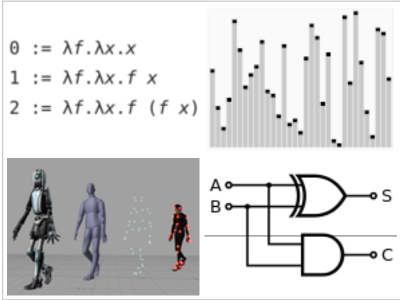
Computer science

Computer science is the study of algorithmic processes, computational machines and computation itself.[1] As a discipline, computer science spans a range of topics from theoretical studies of algorithms, computation and information to the practical issues of implementing computational systems in hardware and software.[2][3]



Computer science deals with the theoretical foundations of information, algorithms and the architectures of its computation as well as practical techniques for their application.

Its fields can be divided into theoretical and practical disciplines. For example, the theory of computation concerns abstract models of computation and general classes of problems that can be solved using them, while computer graphics or computational geometry emphasize more specific applications. Algorithms and data structures have been called the heart of computer science.[4] Programming language theory considers approaches to the description of computational processes, while computer programming involves the use of them to create complex systems. Computer architecture describes construction of computer components and computer-operated equipment. Artificial intelligence aims to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, planning and learning found in humans and animals. A digital computer is capable of simulating various information processes.[5] The fundamental concern of computer science is determining what can and cannot be automated.[6] Computer scientists usually focus on academic research. The Turing Award is generally recognized as the highest distinction in computer sciences.

Contents

History

The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity, aiding in computations such as multiplication and division. Algorithms for performing computations have existed since antiquity, even before the development of sophisticated computing equipment.

Charles Babbage, sometimes referred to as the "father of computing".[7]

Wilhelm Schickard designed and constructed the first working mechanical calculator in 1623.[9] In 1673, Gottfried Leibniz demonstrated a digital mechanical calculator, called the Stepped Reckoner.[10] Leibniz may be considered the first computer scientist and information theorist, for, among other reasons, documenting the binary number system. In 1820, Thomas de Colmar launched the mechanical calculator industry[note 1] when he invented his simplified arithmometer, the first calculating machine strong enough and reliable enough to be used daily in an office environment. Charles Babbage started the design of the first *automatic mechanical calculator*, his Difference Engine, in 1822, which eventually gave him the idea of the first *programmable mechanical calculator*, his Analytical Engine.[11] He started developing this machine in 1834, and "in less than two years, he had sketched out many of the salient features of the modern computer".[12] "A crucial step was the adoption of a punched card system derived from the Jacquard loom"[12] making it infinitely programmable.[note 2] In 1843, during the translation of a French article on the Analytical Engine, Ada Lovelace wrote, in one of the many notes she included, an algorithm to compute the Bernoulli numbers, which is considered to be the first published algorithm ever specifically tailored for implementation on a computer.[13] Around 1885, Herman Hollerith invented the tabulator, which used punched cards to process statistical information; eventually his company became part of IBM. Following Babbage, although unaware of his earlier work, Percy Ludgate in 1909 published [14] the 2nd of the only two designs for mechanical analytical engines in history. In 1937, one hundred years after Babbage's impossible dream, Howard Aiken convinced IBM, which was making all kinds of punched card equipment and was also in the calculator business[15] to develop his giant programmable calculator, the ASCC/Harvard Mark I, based on Babbage's Analytical Engine, which itself used cards and a central computing unit. When the machine was finished, some hailed it as "Babbage's dream come true".[16]

Ada Lovelace published the first algorithm intended for processing on a computer.[8]

During the 1940s, with the development of new and more powerful computing machines such as the Atanasoff–Berry computer and ENIAC, the term *computer* came to refer to the machines rather than their human predecessors.[17] As it became clear that computers could be used for more than just mathematical calculations, the field of computer science broadened to study computation in general. In 1945, IBM founded the Watson Scientific Computing Laboratory at Columbia University in New York City. The renovated fraternity house on Manhattan's West Side was IBM's first laboratory devoted to pure science. The lab is the forerunner of IBM's Research Division, which today operates research facilities around the world.[18] Ultimately, the close relationship between IBM and the university was instrumental in the emergence of a new scientific discipline, with Columbia offering one of the first academic-credit courses in computer science in 1946.[19] Computer science began to be established as a distinct academic discipline in the 1950s and early 1960s.[20][21] The world's first computer science degree program, the Cambridge Diploma in Computer Science, began at the University of Cambridge Computer Laboratory in 1953. The first computer science department in the United States was formed at Purdue University in 1962.[22] Since practical computers became available, many applications of computing have become distinct areas of study in their own rights.

Etymology

Although first proposed in 1956,[23] the term "computer science" appears in a 1959 article in *Communications of the ACM*,[24] in which Louis Fein argues for the creation of a *Graduate School in Computer Sciences* analogous to the creation of Harvard Business School in 1921,[25] justifying the name by arguing that, like management science, the subject is applied and interdisciplinary in nature, while having the characteristics typical of an academic discipline.[24] His efforts, and those of others such as numerical analyst George Forsythe, were rewarded: universities went on to create such departments, starting with Purdue in 1962.[26] Despite its name, a significant amount of computer science does not involve the study of computers themselves. Because of this, several alternative names have been proposed.[27] Certain departments of major universities prefer the term *computing science*, to emphasize precisely that difference. Danish scientist Peter Naur suggested the term *datalogy*,[28] to reflect the fact that the scientific discipline revolves around data and data treatment, while not necessarily involving computers. The first scientific institution to use the term was the Department of Datalogy at the University of Copenhagen, founded in 1969, with Peter Naur being the first professor in datalogy. The term is used mainly in the Scandinavian countries. An alternative term, also proposed by Naur, is data science; this is now used for a multi-disciplinary field of data analysis, including statistics and databases.

In the early days of computing, a number of terms for the practitioners of the field of computing were suggested in the *Communications of the ACM*—*turingineer*, *turologist*, *flow-charts-man*, *applied meta-mathematician*, and *applied epistemologist*.[29] Three months later in the same journal, *comptologist* was suggested, followed next year by *hypologist*.[30] The term *computics* has also been suggested.[31] In Europe, terms derived from contracted translations of the expression "automatic information" (e.g. "informazione automatica" in Italian) or "information and mathematics" are often used, e.g. *informatique* (French), *Informatik* (German), *informatica* (Italian, Dutch), *informática* (Spanish, Portuguese), *informatika* (Slavic languages and Hungarian) or *pliroforiki* (*πληροφορική*, which means informatics) in Greek. Similar words have also been adopted in the UK (as in *the School of Informatics of the University of Edinburgh*).[32] "In the U.S., however, informatics is linked with applied computing, or computing in the context of another domain."[33]

A folkloric quotation, often attributed to—but almost certainly not first formulated by—Edsger Dijkstra, states that "computer science is no more about computers than astronomy is about telescopes."[note 3] The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of computer hardware is usually considered part of computer engineering, while the study of commercial computer systems and their deployment is often called information technology or information systems. However, there has been much cross-fertilization of ideas between the various computer-related disciplines. Computer science research also often intersects other disciplines, such as philosophy, cognitive science, linguistics, mathematics, physics, biology, Earth science, statistics, and logic.

Computer science is considered by some to have a much closer relationship with mathematics than many scientific disciplines, with some observers saying that computing is a mathematical science.[20] Early computer science was strongly influenced by the work of mathematicians such as Kurt Gödel, Alan Turing, John von Neumann, Rózsa Péter and Alonzo Church and there continues to be a useful interchange of ideas between the two fields in areas such as mathematical logic, category theory, domain theory, and algebra.[23]

The relationship between Computer Science and Software Engineering is a contentious issue, which is further muddied by disputes over what the term "Software Engineering" means, and how computer science is defined.[34] David Parnas, taking a cue from the relationship between other engineering and science disciplines, has claimed that the principal focus of computer science is studying the properties of computation in general, while the principal focus of software engineering is the design of specific computations to achieve practical goals, making the two separate but complementary disciplines.[35]

The academic, political, and funding aspects of computer science tend to depend on whether a department is formed with a mathematical emphasis or with an engineering emphasis. Computer science departments with a mathematics emphasis and with a numerical orientation consider alignment with computational science. Both types of departments tend to make efforts to bridge the field educationally if not across all research.

Philosophy

A number of computer scientists have argued for the distinction of three separate paradigms in computer science. Peter Wegner argued that those paradigms are science, technology, and mathematics.[36] Peter Denning's working group argued that they are theory, abstraction (modeling), and design.[37] Amnon H. Eden described them as the "rationalist paradigm" (which treats computer science as a branch of mathematics, which is prevalent in theoretical computer science, and mainly employs deductive reasoning), the "technocratic paradigm" (which might be found in engineering approaches, most prominently in software engineering), and the "scientific paradigm" (which approaches computer-related artifacts from the empirical perspective of natural sciences, identifiable in some branches of artificial intelligence).[38] Computer science focuses on methods involved in design, specification, programming, verification, implementation and testing of human-made computing systems.[39]

Fields

> Computer science is no more about computers than astronomy is about telescopes.
>
> — Edsger Dijkstra

As a discipline, computer science spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software.[40][41] CSAB, formerly called Computing Sciences Accreditation Board—which is made up of representatives of the Association for Computing Machinery (ACM), and the IEEE Computer Society (IEEE CS)[42]—identifies four areas that it considers crucial to the discipline of computer science: *theory of computation*, *algorithms and data structures*, *programming methodology and languages*, and *computer elements and architecture*. In addition to these four areas, CSAB also identifies fields such as software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, human–computer interaction, computer graphics, operating systems, and numerical and symbolic computation as being important areas of computer science.[40]

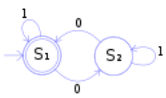
Theoretical computer science

Theoretical Computer Science is mathematical and abstract in spirit, but it derives its motivation from the practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

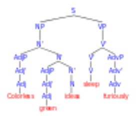
Theory of computation

According to Peter Denning, the fundamental question underlying computer science is, "What can be automated?"[20] Theory of computation is focused on answering fundamental questions about what can be computed and what amount of resources are required to perform those computations. In an effort to answer the first question, computability theory examines which computational problems are solvable on various theoretical models of computation. The second question is addressed by computational complexity theory, which studies the time and space costs associated with different approaches to solving a multitude of computational problems.

The famous P = NP? problem, one of the Millennium Prize Problems,[43] is an open problem in the theory of computation.



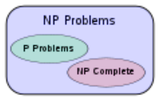
Automata theory



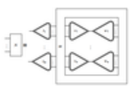
Formal languages

$$M = \{X : X \notin X\}$$

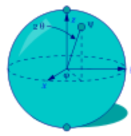
Computability theory



Computational complexity theory

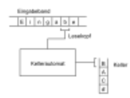


Models of computation



Quantum computing theory

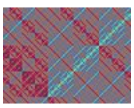
Logic circuit theory



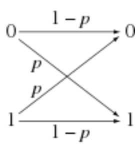
Cellular automata

Information and coding theory

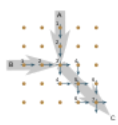
Information theory, closely related to probability and statistics, is related to the quantification of information. This was developed by Claude Shannon to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data.[44] Coding theory is the study of the properties of codes (systems for converting information from one form to another) and their fitness for a specific application. Codes are used for data compression, cryptography, error detection and correction, and more recently also for network coding. Codes are studied for the purpose of designing efficient and reliable data transmission methods. [45]



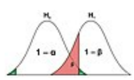
Coding theory



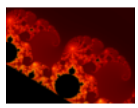
Channel capacity



Algorithmic information theory



Signal detection theory



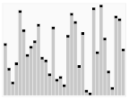
Kolmogorov complexity

Data structures and algorithms

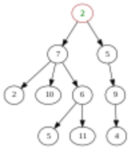
Data structures and algorithms are the studies of commonly used computational methods and their computational efficiency.

$O(n^2)$

Analysis of algorithms



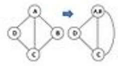
Algorithm design



Data structures

Combinatorial optimization

Computational geometry

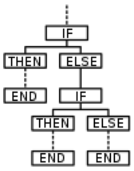


Randomized algorithms

Programming language theory and formal methods

Programming language theory is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages and their individual features. It falls within the discipline of computer science, both depending on and affecting mathematics, software engineering, and linguistics. It is an active research area, with numerous dedicated academic journals.

Formal methods are a particular kind of mathematically based technique for the specification, development and verification of software and hardware systems.[46] The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design. They form an important theoretical underpinning for software engineering, especially where safety or security is involved. Formal methods are a useful adjunct to software testing since they help avoid errors and can also give a framework for testing. For industrial use, tool support is required. However, the high cost of using formal methods means that they are usually only used in the development of high-integrity and life-critical systems, where safety or security is of utmost importance. Formal methods are best described as the application of a fairly broad variety of theoretical computer science fundamentals, in particular logic calculi, formal languages, automata theory, and program semantics, but also type systems and algebraic data types to problems in software and hardware specification and verification.



Formal semantics

$\Gamma \vdash x : \mathrm{Int}$

Type theory

Compiler design

Programming languages

$(a\vee\neg b)\wedge b$

$\neg a$

Formal verification

Automated theorem proving

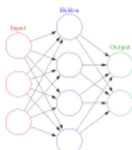
Computer systems and computational processes

Artificial intelligence

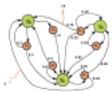
Artificial intelligence (AI) aims to or is required to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, learning, and communication found in humans and animals. From its origins in cybernetics and in the Dartmouth Conference (1956), artificial intelligence research has been necessarily cross-disciplinary, drawing on areas of expertise such as applied mathematics, symbolic logic, semiotics, electrical engineering, philosophy of mind, neurophysiology, and social intelligence. AI is associated in the popular mind with robotic development, but the main field of practical application has been as an embedded component in areas of software development, which require computational understanding. The starting point in the late 1940s was Alan Turing's question "Can computers think?", and the question remains effectively unanswered, although the Turing test is still used to assess computer output on the scale of human intelligence. But the automation of evaluative and predictive tasks has been increasingly successful as a substitute for human monitoring and intervention in domains of computer application involving complex real-world data.

Computational learning theory

Computer vision



Neural networks

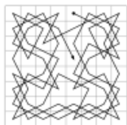


Planning and scheduling

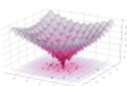
abcdefghijk
lmnopqrstuvwxyz

ABCDEFGHIJKLMN
OPQRSTUVWXYZ

Natural language processing

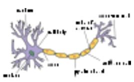


Computational game theory

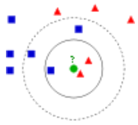


Evolutionary computation

Autonomic computing

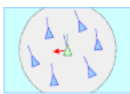


Representation and reasoning



Pattern recognition

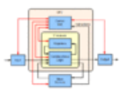
Robotics



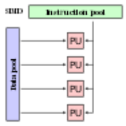
Swarm intelligence

Computer architecture and organization

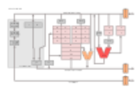
Computer architecture, or digital computer organization, is the conceptual design and fundamental operational structure of a computer system. It focuses largely on the way by which the central processing unit performs internally and accesses addresses in memory.[47] Computer engineers study computational logic and design of computer hardware, from individual processor components, microcontrollers, personal computers to supercomputers and embedded systems. The term "architecture" in computer literature can be traced to the work of Lyle R. Johnson and Frederick P. Brooks, Jr., members of the Machine Organization department in IBM's main research center in 1959.


Processing unit


Microarchitecture

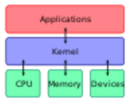


Multiprocessing


Processor design


Ubiquitous computing

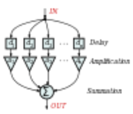

Systems architecture



Operating systems

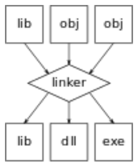

Input/output


Embedded system



Real-time computing


Dependability



Interpreter

Concurrent, parallel and distributed computing

Concurrency is a property of systems in which several computations are executing simultaneously, and potentially interacting with each other.[48] A number of mathematical models have been developed for general concurrent computation including Petri nets, process calculi and the Parallel Random Access Machine model.[49] When multiple computers are connected in a network while using concurrency, this is known as a distributed system. Computers within that distributed system have their own private memory, and information can be exchanged to achieve common goals.[50]

Computer networks

This branch of computer science aims to manage networks between computers worldwide.

Computer security and cryptography

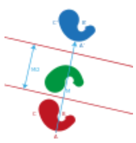
Computer security is a branch of computer technology with the objective of protecting information from unauthorized access, disruption, or modification while maintaining the accessibility and usability of the system for its intended users. Cryptography is the practice and study of hiding (encryption) and therefore deciphering (decryption) information. Modern cryptography is largely related to computer science, for many encryption and decryption algorithms are based on their computational complexity.

Databases and data mining

A database is intended to organize, store, and retrieve large amounts of data easily. Digital databases are managed using database management systems to store, create, maintain, and search data, through database models and query languages. Data mining is a process of discovering patterns in large data sets.

Computer graphics and visualization

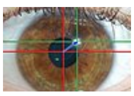
Computer graphics is the study of digital visual contents and involves the synthesis and manipulation of image data. The study is connected to many other fields in computer science, including computer vision, image processing, and computational geometry, and is heavily applied in the fields of special effects and video games.


2D computer graphics


Computer animation


Rendering


Mixed reality


Virtual reality

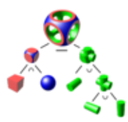
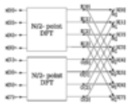

Solid modeling

Image and sound processing

Information can take the form of images, sound, video or other multimedia. Bits of information can be streamed via signals. Its processing is the central notion of informatics, the European view on computing, which studies information processing algorithms independently of the type of information carrier - whether it is electrical, mechanical or biological. This field plays important role in information theory, telecommunications, information engineering and has applications in medical image computing and speech synthesis, among others. *What is the lower bound on the complexity of fast Fourier transform algorithms?* is one of unsolved problems in theoretical computer science.


FFT algorithms

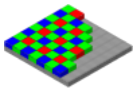
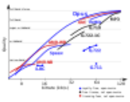
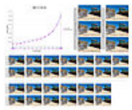
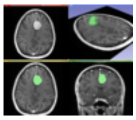


Image processing


Speech recognition


Data compression


Medical image computing


Speech synthesis

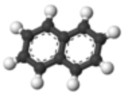
Applied computer science

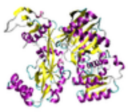
Computational science, finance and engineering

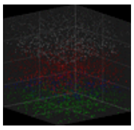
Scientific computing (or computational science) is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems. A major usage of scientific computing is simulation of various processes, including computational fluid dynamics, physical, electrical, and electronic systems and circuits, as well as societies and social situations (notably war games) along with their habitats, among many others. Modern computers enable optimization of such designs as complete aircraft. Notable in electrical and electronic circuit design are SPICE,[51] as well as software for physical realization of new (or modified) designs. The latter includes essential design software for integrated circuits.


Numerical analysis


Computational physics

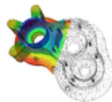

Computational chemistry


Bioinformatics


Neuroinformatics


Psychoinformatics


Medical informatics


Computational engineering


Comput music

Social computing and human-computer interaction

Social computing is an area that is concerned with the intersection of social behavior and computational systems. Human-computer interaction research develops theories, principles, and guidelines for user interface designers.

Software engineering

Software engineering is the study of designing, implementing, and modifying the software in order to ensure it is of high quality, affordable, maintainable, and fast to build. It is a systematic approach to software design, involving the application of engineering practices to software. Software engineering deals with the organizing and analyzing of software—it doesn't just deal with the creation or manufacture of new software, but its internal arrangement and maintenance. For example software testing, systems engineering, technical debt and software development processes.

Discoveries

The philosopher of computing Bill Rapaport noted three *Great Insights of Computer Science*:[52]

- Gottfried Wilhelm Leibniz's, George Boole's, Alan Turing's, Claude Shannon's, and Samuel Morse's insight: there are only *two objects* that a computer has to deal with in order to represent "anything".[note 4]

  All the information about any computable problem can be represented using only 0 and 1 (or any other bistable pair that can flip-flop between two easily distinguishable states, such as "on/off", "magnetized/de-magnetized", "high-voltage/low-voltage", etc.).

- Alan Turing's insight: there are only *five actions* that a computer has to perform in order to do "anything".

  Every algorithm can be expressed in a language for a computer consisting of only five basic instructions:[53]

  - move left one location;
  - move right one location;
  - read symbol at current location;
  - print 0 at current location;
  - print 1 at current location.

- Corrado Böhm and Giuseppe Jacopini's insight: there are only *three ways of combining* these actions (into more complex ones) that are needed in order for a computer to do "anything".[54]

  Only three rules are needed to combine any set of basic instructions into more complex ones:

  - *sequence*: first do this, then do that;
  - *selection*: IF such-and-such is the case, THEN do this, ELSE do that;

- *repetition*: WHILE such-and-such is the case, DO this.

Note that the three rules of Boehm's and Jacopini's insight can be further simplified with the use of goto (which means it is more elementary than structured programming).

Programming paradigms

Programming languages can be used to accomplish different tasks in different ways. Common programming paradigms include:

- Functional programming, a style of building the structure and elements of computer programs that treats computation as the evaluation of mathematical functions and avoids state and mutable data. It is a declarative programming paradigm, which means programming is done with expressions or declarations instead of statements.[55]
- Imperative programming, a programming paradigm that uses statements that change a program's state.[56] In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform. Imperative programming focuses on describing how a program operates.
- Object-oriented programming, a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated. Thus object-oriented computer programs are made out of objects that interact with one another.[57]
- Service-oriented programming, a programming paradigm that uses "services" as the unit of computer work, to design and implement integrated business applications and mission critical software programs

Many languages offer support for multiple paradigms, making the distinction more a matter of style than of technical capabilities.[58]

Academia

Conferences are important events for computer science research. During these conferences, researchers from the public and private sectors present their recent work and meet. Unlike in most other academic fields, in computer science, the prestige of conference papers is greater than that of journal publications.[59][60] One proposed explanation for this is the quick development of this relatively new field requires rapid review and distribution of results, a task better handled by conferences than by journals.[61]

Education

Computer Science, known by its near synonyms, **Computing**, **Computer Studies**, has been taught in UK schools since the days of batch processing, mark sensitive cards and paper tape but usually to a select few students.[62] In 1981, the BBC produced a micro-computer and classroom network and Computer Studies became common for GCE O level students (11–16-year-old), and Computer Science to A level students. Its importance was recognised, and it became a compulsory part of the National Curriculum, for Key Stage 3 & 4. In September 2014 it became an entitlement for all pupils over the age of 4.[63]

In the US, with 14,000 school districts deciding the curriculum, provision was fractured.[64] According to a 2010 report by the Association for Computing Machinery (ACM) and Computer Science Teachers Association (CSTA), only 14 out of 50 states have adopted significant education standards for high school computer science.[65]

Israel, New Zealand, and South Korea have included computer science in their national secondary education curricula,[66][67] and several others are following.[68]

See also

- Computer Science and Engineering
- Computer engineering
- Information technology
- List of computer scientists
- List of computer science awards
- List of important publications in computer science
- List of pioneers in computer science
- List of unsolved problems in computer science
- List of terms relating to algorithms and data structures
- Digital Revolution
- Software engineering
- Programming language
- Algorithmic trading
- Information and communications technology

Notes

1. In 1851
2. "The introduction of punched cards into the new engine was important not only as a more convenient form of control than the drums, or because programs could now be of unlimited extent, and could be stored and repeated without the danger of introducing errors in setting the machine by hand; it was important also because it served to crystallize Babbage's feeling that he had invented something really new, something much more than a sophisticated calculating machine." Bruce Collier, 1970
3. See the entry "Computer science" on Wikiquote for the history of this quotation.
4. The word "anything" is written in quotation marks because there are things that computers cannot do. One example is: to answer the question if an arbitrary given computer program will eventually finish or run forever (the Halting problem).

References

1. "What is Computer Science? - Computer Science.The University of York" (https://www.cs.york.ac.uk/undergraduate/what-is-cs/). *www.cs.york.ac.uk*. Retrieved June 11, 2020.
2. "WordNet Search—3.1" (http://wordnetweb.princeton.edu/perl/webwn?s=computer%20scientist). Wordnetweb.princeton.edu. Retrieved May 14, 2012.
3. "Definition of computer science | Dictionary.com" (https://www.dictionary.com/browse/computer-science). *www.dictionary.com*. Retrieved June 11, 2020.
4. Harel, David. (2014). *Algorithmics The Spirit of Computing* (http://worldcat.org/oclc/876384882). Springer Berlin. ISBN 978-3-642-44135-6. OCLC 876384882 (https://www.worldcat.org/oclc/876384882).

5. "COMPUTER SCIENCE: THE DISCIPLINE" (http://www.idi.ntnu.no/emner/dif8916/denning.pdf) (PDF). May 25, 2006. Archived (https://web.archive.org/web/20060525195404/http://www.idi.ntnu.no/emner/dif8916/denning.pdf) (PDF) from the original on May 25, 2006. Retrieved January 4, 2021.
6. The MIT Press. "What Can Be Automated? Computer Science and Engineering Research Study | The MIT Press" (https://mitpress.mit.edu/books/what-can-be-automated). *mitpress.mit.edu*.
7. "Charles Babbage Institute: Who Was Charles Babbage?" (http://www.cbi.umn.edu/about/babbage.html). *cbi.umn.edu*. Retrieved December 28, 2016.
8. "Ada Lovelace | Babbage Engine | Computer History Museum" (http://www.computerhistory.org/babbage/adalovelace/). *www.computerhistory.org*. Retrieved December 28, 2016.
9. "Wilhelm Schickard – Ein Computerpionier" (http://www.fmi.uni-jena.de/fmimedia/Fakultaet/Institute+und+Abteilungen/Abteilung+f%C3%BCr+Didaktik/GDI/Wilhelm+Schickard.pdf) (PDF) (in German).
10. Keates, Fiona (June 25, 2012). "A Brief History of Computing" (http://blogs.royalsociety.org/history-of-science/2012/06/25/history-of-computing/). *The Repository*. The Royal Society.
11. "Science Museum, Babbage's Analytical Engine, 1834-1871 (Trial model)" (https://collection.sciencemuseumgroup.org.uk/objects/co62245/babbages-analytical-engine-1834-1871-trial-model-analytical-engines). Retrieved May 11, 2020.
12. Anthony Hyman (1982). *Charles Babbage, pioneer of the computer* (https://archive.org/details/charlesbabbagepi0000hyma).
13. "A Selection and Adaptation From Ada's Notes found in Ada, The Enchantress of Numbers," by Betty Alexandra Toole Ed.D. Strawberry Press, Mill Valley, CA" (https://web.archive.org/web/20060210172109/http://www.scottlan.edu/lriddle/women/ada-love.htm). Archived from the original (http://www.scottlan.edu/Lriddle/women/ada-love.htm) on February 10, 2006. Retrieved May 4, 2006.
14. "The John Gabriel Byrne Computer Science Collection" (https://web.archive.org/web/20190416071721/https://www.scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/TCD-SCSS-X.20121208.002.pdf) (PDF). Archived from the original (https://scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/TCD-SCSS-X.20121208.002.pdf/) on April 16, 2019. Retrieved August 8, 2019.
15. "In this sense Aiken needed IBM, whose technology included the use of punched cards, the accumulation of numerical data, and the transfer of numerical data from one register to another", Bernard Cohen, p.44 (2000)
16. Brian Randell, p. 187, 1975
17. The Association for Computing Machinery (ACM) was founded in 1947.
18. "IBM Archives: 1945" (https://www.ibm.com/ibm/history/history/year_1945.html). Ibm.com. Retrieved March 19, 2019.
19. "IBM100 – The Origins of Computer Science" (https://www.ibm.com/ibm/history/ibm100/us/en/icons/compsci/). Ibm.com. September 15, 1995. Retrieved March 19, 2019.
20. Denning, Peter J. (2000). "Computer Science: The Discipline" (https://web.archive.org/web/20060525195404/http://www.idi.ntnu.no/emner/dif8916/denning.pdf) (PDF). *Encyclopedia of Computer Science*. Archived from the original (http://www.idi.ntnu.no/emner/dif8916/denning.pdf) (PDF) on May 25, 2006.
21. "Some EDSAC statistics" (http://www.cl.cam.ac.uk/conference/EDSAC99/statistics.html). University of Cambridge. Retrieved November 19, 2011.
22. "Computer science pioneer Samuel D. Conte dies at 85" (http://www.cs.purdue.edu/about/conte.html). Purdue Computer Science. July 1, 2002. Retrieved December 12, 2014.
23. Tedre, Matti (2014). *The Science of Computing: Shaping a Discipline*. Taylor and Francis / CRC Press.
24. Louis Fine (1959). "The Role of the University in Computers, Data Processing, and Related Fields". *Communications of the ACM*. **2** (9): 7–14. doi:10.1145/368424.368427 (https://doi.org/10.1145%2F368424.368427). S2CID 6740821 (https://api.semanticscholar.org/CorpusID:6740821).
25. "Stanford University Oral History" (http://library.stanford.edu/guides/stanford-university-oral-history). Stanford University. Retrieved May 30, 2013.
26. Donald Knuth (1972). *"George Forsythe and the Development of Computer Science" (http://www.stanford.edu/dept/ICME/docs/history/forsythe_knuth.pdf)*. *Comms. ACM*. Archived (https://web.archive.org/web/20131020200802/http://www.stanford.edu/dept/ICME/docs/history/forsythe_knuth.pdf) October 20, 2013, at the Wayback Machine
27. Matti Tedre (2006). "The Development of Computer Science: A Sociocultural Perspective" (http://epublications.uef.fi/pub/urn_isbn_952-458-867-6/urn_isbn_952-458-867-6.pdf) (PDF). p. 260. Retrieved December 12, 2014.
28. Peter Naur (1966). "The science of datalogy". *Communications of the ACM*. **9** (7): 485. doi:10.1145/365719.366510 (https://doi.org/10.1145%2F365719.366510). S2CID 47558402 (https://api.semanticscholar.org/CorpusID:47558402).
29. Weiss, E.A.; Corley, Henry P.T. "Letters to the editor". *Communications of the ACM*. **1** (4): 6. doi:10.1145/368796.368802 (https://doi.org/10.1145%2F368796.368802). S2CID 5379449 (https://api.semanticscholar.org/CorpusID:5379449).
30. Communications of the ACM 2(1):p.4
31. IEEE Computer 28(12): p.136
32. P. Mounier-Kuhn, *L'Informatique en France, de la seconde guerre mondiale au Plan Calcul. L'émergence d'une science*, Paris, PUPS, 2010, ch. 3 & 4.
33. Groth, Dennis P. (February 2010). "Why an Informatics Degree?" (http://cacm.acm.org/magazines/2010/2/69363-why-an-informatics-degree). *Communications of the ACM*. Cacm.acm.org.
34. Tedre, M. (2011). "Computing as a Science: A Survey of Competing Viewpoints". *Minds and Machines*. **21** (3): 361–387. doi:10.1007/s11023-011-9240-4 (https://doi.org/10.1007%2Fs11023-011-9240-4). S2CID 14263916 (https://api.semanticscholar.org/CorpusID:14263916).
35. Parnas, D.L. (1998). "Software engineering programmes are not computer science programmes". *Annals of Software Engineering*. **6**: 19–37. doi:10.1023/A:1018949113292 (https://doi.org/10.1023%2FA%3A1018949113292). S2CID 35786237 (https://api.semanticscholar.org/CorpusID:35786237)., p. 19: "Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering, […]"
36. Wegner, P. (October 13–15, 1976). *Research paradigms in computer science—Proceedings of the 2nd international Conference on Software Engineering*. San Francisco, California, United States: IEEE Computer Society Press, Los Alamitos, CA.
37. Denning, P.J.; Comer, D.E.; Gries, D.; Mulder, M.C.; Tucker, A.; Turner, A.J.; Young, P.R. (January 1989). "Computing as a discipline". *Communications of the ACM*. **32**: 9–23. doi:10.1145/63238.63239 (https://doi.org/10.1145%2F63238.63239). S2CID 723103 (https://api.semanticscholar.org/CorpusID:723103).
38. Eden, A.H. (2007). "Three Paradigms of Computer Science" (https://web.archive.org/web/20160215100211/http://www.eden-study.org/articles/2007/three_paradigms_of_computer_science.pdf) (PDF). *Minds and Machines*. **17** (2): 135–167. CiteSeerX 10.1.1.304.7763 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.304.7763). doi:10.1007/s11023-007-9060-8 (https://doi.org/10.1007%2Fs11023-007-9060-8). S2CID 3023076 (https://api.semanticscholar.org/CorpusID:3023076). Archived from the original (http://www.eden-study.org/articles/2007/three_paradigms_of_computer_science.pdf) (PDF) on February 15, 2016.
39. Turner, Raymond; Angius, Nicola (2019). "The Philosophy of Computer Science" (https://plato.stanford.edu/archives/spr2019/entries/computer-science/). In Zalta, Edward N. (ed.). *The Stanford Encyclopedia of Philosophy*.
40. "Computer Science as a Profession" (https://web.archive.org/web/20080617030847/http://www.csab.org/comp_sci_profession.html). Computing Sciences Accreditation Board. May 28, 1997. Archived from the original (http://www.csab.org/comp_sci_profession.html) on June 17, 2008. Retrieved May 23, 2010.
41. Committee on the Fundamentals of Computer Science: Challenges and Opportunities, National Research Council (2004). *Computer Science: Reflections on the Field, Reflections from the Field* (http://www.nap.edu/catalog.php?record_id=11106#toc). National Academies Press. ISBN 978-0-309-09301-9.
42. "CSAB Leading Computer Education" (http://www.csab.org/). CSAB. August 3, 2011. Retrieved November 19, 2011.
43. Clay Mathematics Institute (http://www.claymath.org/millennium/P_vs_NP/) P = NP Archived (https://web.archive.org/web/20131014194456/http://www.claymath.org/millennium/P_vs_NP/) October 14, 2013, at the Wayback Machine
44. P. Collins, Graham (October 14, 2002). "Claude E. Shannon: Founder of Information Theory" (http://www.scientificamerican.com/article.cfm?id=claude-e-shannon-founder). *Scientific American*. Retrieved December 12, 2014.
45. Van-Nam Huynh; Vladik Kreinovich; Songsak Sriboonchitta; 2012. Uncertainty Analysis in Econometrics with Applications. Springer Science & Business Media. p. 63. ISBN 978-3-642-35443-4.

46. Phillip A. Laplante, 2010. Encyclopedia of Software Engineering Three-Volume Set (Print). CRC Press. p. 309. ISBN 978-1-351-24926-3.
47. A. Thisted, Ronald (April 7, 1997). "Computer Architecture" (http://galton.uchicago.edu/~thisted/Distribute/comparch.pdf) (PDF). The University of Chicago.
48. Jiacun Wang, 2017. Real-Time Embedded Systems. Wiley. p. 12. ISBN 978-1-119-42070-5.
49. Gordana Dodig-Crnkovic; Raffaela Giovagnoli; 2013. Computing Nature: Turing Centenary Perspective. Springer Science & Business Media. p. 247. ISBN 978-3-642-37225-4.
50. Simon Elias Bibri; 2018. Smart Sustainable Cities of the Future: The Untapped Potential of Big Data Analytics and Context-Aware Computing for Advancing Sustainability. Springer. p. 74. ISBN 978-3-319-73981-6.
51. Muhammad H. Rashid, 2016. SPICE for Power Electronics and Electric Power. CRC Press. p. 6. ISBN 978-1-4398-6047-2.
52. Rapaport, William J. (September 20, 2013). "What Is Computation?" (http://www.cse.buffalo.edu/~rapaport/computation.html). State University of New York at Buffalo.
53. B. Jack Copeland, 2012. Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer. OUP Oxford. p. 107. ISBN 978-0-19-960915-4.
54. Charles W. Herbert, 2010. An Introduction to Programming Using Alice 2.2. Cengage Learning. p. 122. ISBN 0-538-47866-7.
55. Md. Rezaul Karim; Sridhar Alla; 2017. Scala and Spark for Big Data Analytics: Explore the concepts of functional programming, data streaming, and machine learning. Packt Publishing Ltd. p. 87. ISBN 978-1-78355-050-0.
56. Lex Sheehan, 2017. Learning Functional Programming in Go: Change the way you approach your applications using functional programming in Go. Packt Publishing Ltd. p. 16. ISBN 978-1-78728-604-7.
57. Evelio Padilla, 2015. Substation Automation Systems: Design and Implementation. Wiley. p. 245. ISBN 978-1-118-98730-8.
58. "Multi-Paradigm Programming Language" (https://web.archive.org/web/20130821052407/https://developer.mozilla.org/en-US/docs/multiparadigmlanguage.html). *developer.mozilla.org*. Mozilla Foundation. Archived from the original (https://developer.mozilla.org/en-US/docs/multiparadigmlanguage.html) on August 21, 2013.
59. Meyer, Bertrand (April 2009). "Viewpoint: Research evaluation for computer science". *Communications of the ACM*. **25** (4): 31–34. doi:10.1145/1498765.1498780 (https://doi.org/10.1145%2F1498765.1498780). S2CID 8625066 (https://api.semanticscholar.org/CorpusID:8625066).
60. Patterson, David (August 1999). "Evaluating Computer Scientists and Engineers For Promotion and Tenure" (http://cra.org/resources/bp-view/evaluating_computer_scientists_and_engineers_for_promotion_and_tenure/). Computing Research Association.
61. Fortnow, Lance (August 2009). "Viewpoint: Time for Computer Science to Grow Up" (http://cacm.acm.org/magazines/2009/8/34492-viewpoint-time-for-computer-science-to-grow-up/fulltext). *Communications of the ACM*. **52** (8): 33–35. doi:10.1145/1536616.1536631 (https://doi.org/10.1145%2F1536616.1536631).
62. Burns, Judith (April 3, 2016). "Computer science A-level 1970s style" (https://www.bbc.co.uk/news/education-35890450). Retrieved February 9, 2019.
63. Jones, Michael (October 1915). "Developing a Computer Science Curriculum in England: Exploring Approaches in the USA" (https://www.wcmt.org.uk/sites/default/files/report-documents/Jones%20M%20Report%202015%20%20Final.pdf) (PDF). Winston Churchill Memorial Trust. Retrieved February 9, 2019.
64. "Computer Science: Not Just an Elective Anymore" (http://www.edweek.org/ew/articles/2014/02/26/22computer_ep.h33.html). *Education Week*. February 25, 2014.
65. Wilson, Cameron; Sudol, Leigh Ann; Stephenson, Chris; Stehlik, Mark (2010). "Running on Empty: The Failure to Teach K–12 Computer Science in the Digital Age" (http://runningonempty.acm.org/fullreport2.pdf) (PDF). ACM.
66. "A is for algorithm" (https://www.economist.com/news/international/21601250-global-push-more-computer-science-classrooms-starting-bear-fruit). *The Economist*. April 26, 2014.
67. "Computing at School International comparisons" (http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf) (PDF). Retrieved July 20, 2015.
68. "Adding Coding to the Curriculum" (https://www.nytimes.com/2014/03/24/world/europe/adding-coding-to-the-curriculum.html). *The New York Times*. March 23, 2014.

Further reading

Overview

- Tucker, Allen B. (2004). *Computer Science Handbook* (2nd ed.). Chapman and Hall/CRC. ISBN 978-1-58488-360-9.
  - "Within more than 70 chapters, every one new or significantly revised, one can find any kind of information and references about computer science one can imagine. […] all in all, there is absolute nothing about Computer Science that can not be found in the 2.5 kilogram-encyclopaedia with its 110 survey articles […]." (Christoph Meinel, *Zentralblatt MATH*)
- van Leeuwen, Jan (1994). *Handbook of Theoretical Computer Science*. The MIT Press. ISBN 978-0-262-72020-5.
  - "[…] this set is the most unique and possibly the most useful to the [theoretical computer science] community, in support both of teaching and research […]. The books can be used by anyone wanting simply to gain an understanding of one of these areas, or by someone desiring to be in research in a topic, or by instructors wishing to find timely information on a subject they are teaching outside their major areas of expertise." (Rocky Ross, *SIGACT News*)
- Ralston, Anthony; Reilly, Edwin D.; Hemmendinger, David (2000). *Encyclopedia of Computer Science* (http://portal.acm.org/ralston.cfm) (4th ed.). Grove's Dictionaries. ISBN 978-1-56159-248-7.
  - "Since 1976, this has been the definitive reference work on computer, computing, and computer science. […] Alphabetically arranged and classified into broad subject areas, the entries cover hardware, computer systems, information and data, software, the mathematics of computing, theory of computation, methodologies, applications, and computing milieu. The editors have done a commendable job of blending historical perspective and practical reference information. The encyclopedia remains essential for most public and academic library reference collections." (Joe Accardin, Northeastern Illinois Univ., Chicago)
- Edwin D. Reilly (2003). *Milestones in Computer Science and Information Technology* (https://archive.org/details/milestonesincomp0000reil). Greenwood Publishing Group. ISBN 978-1-57356-521-9.

Selected literature

- Knuth, Donald E. (1996). *Selected Papers on Computer Science*. CSLI Publications, Cambridge University Press.
- Collier, Bruce (1990). *The little engine that could've: The calculating machines of Charles Babbage* (http://robroy.dyndns.info/collier/index.html). Garland Publishing Inc. ISBN 978-0-8240-0043-1.
- Cohen, Bernard (2000). *Howard Aiken, Portrait of a computer pioneer*. The MIT press. ISBN 978-0-262-53179-5.
- Tedre, Matti (2014). *The Science of Computing: Shaping a Discipline*. CRC Press, Taylor & Francis.
- Randell, Brian (1973). *The origins of Digital computers, Selected Papers*. Springer-Verlag. ISBN 978-3-540-06169-4.

- "Covering a period from 1966 to 1993, its interest lies not only in the content of each of these papers – still timely today – but also in their being put together so that ideas expressed at different times complement each other nicely." (N. Bernard, *Zentralblatt MATH*)

Articles

- Peter J. Denning. *Is computer science science? (http://portal.acm.org/citation.cfm?id=1053309&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618)*, Communications of the ACM, April 2005.
- Peter J. Denning, *Great principles in computing curricula (http://portal.acm.org/citation.cfm?id=971303&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618)*, Technical Symposium on Computer Science Education, 2004.
- Research evaluation for computer science, Informatics Europe report (http://www.eqanie.eu/media/Como%20Conference/Tanca-Research_Assessment_A_new_Initiative_by_Informatics_Europe.pdf) Archived (https://web.archive.org/web/20171018181136/http://www.eqanie.eu/media/Como%20Conference/Tanca-Research_Assessment_A_new_Initiative_by_Informatics_Europe.pdf) October 18, 2017, at the Wayback Machine. Shorter journal version: Bertrand Meyer, Christine Choppy, Jan van Leeuwen and Jorgen Staunstrup, *Research evaluation for computer science*, in Communications of the ACM, vol. 52, no. 4, pp. 31–34, April 2009.

Curriculum and classification

- Association for Computing Machinery. 1998 ACM Computing Classification System (https://web.archive.org/web/20080828002940/http://www.acm.org/class/1998/overview.html). 1998.
- Joint Task Force of Association for Computing Machinery (ACM), Association for Information Systems (AIS) and IEEE Computer Society (IEEE CS). Computing Curricula 2005: The Overview Report (https://web.archive.org/web/20141021153204/http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf). September 30, 2005.
- Norman Gibbs, Allen Tucker. "A model curriculum for a liberal arts degree in computer science". *Communications of the ACM*, Volume 29 Issue 3, March 1986.

External links

- Computer science (https://curlie.org/Computers/Computer_Science/) at Curlie
- Scholarly Societies in Computer Science (http://www.lib.uwaterloo.ca/society/compsci_soc.html)
- What is Computer Science? (https://www.youtube.com/watch?v=fjMU-km-Cso)
- Best Papers Awards in Computer Science since 1996 (http://jeffhuang.com/best_paper_awards.html)
- Photographs of computer scientists (http://se.ethz.ch/~meyer/gallery/) by Bertrand Meyer
- EECS.berkeley.edu (http://www.eecs.berkeley.edu/department/history.shtml)

Bibliography and academic search engines

- CiteSeer^x (http://citeseerx.ist.psu.edu/) (article): search engine, digital library and repository for scientific and academic papers with a focus on computer and information science.
- DBLP Computer Science Bibliography (http://dblp.uni-trier.de/) (article): computer science bibliography website hosted at Universität Trier, in Germany.
- The Collection of Computer Science Bibliographies (http://liinwww.ira.uka.de/bibliography/) (Collection of Computer Science Bibliographies)

Professional organizations

- Association for Computing Machinery (http://www.acm.org/)
- IEEE Computer Society (http://www.computer.org/)
- Informatics Europe (http://www.informatics-europe.org/)
- AAAI (http://www.aaai.org/home.html)
- AAAS Computer Science (https://web.archive.org/web/20160205000119/http://membercentral.aaas.org/categories/computer-science)

Misc

- Computer Science—Stack Exchange (https://cs.stackexchange.com/): a community-run question-and-answer site for computer science
- What is computer science (http://www.cs.bu.edu/AboutCS/WhatIsCS.pdf)
- Is computer science science? (https://web.archive.org/web/20170810205524/https://www.cs.mtu.edu/~john/jenning.pdf)
- Computer Science (Software) Must be Considered as an Independent Discipline. (https://www.researchgate.net/publication/306078165_Computer_Science_Software_Must_be_Considered_as_an_Independent_Discipline_Computer_Science_Software_must_not_be_Treated_as_a_Sub-Domain_or_Subset_of_Mathematics)