Computer programming

computerProgramming.doc

Computer programming (often shortened to **programming** or **coding**) is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs. This source code is written in a programming language. The code may be a modification of an existing source or something completely new. The purpose of programming is to create a program that exhibits a certain desired behavior (customization). The process of writing source code requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

Overview

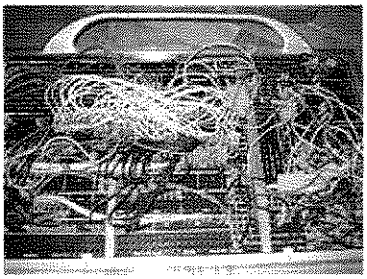
Within software engineering, programming (the *implementation*) is regarded as one phase in a software development process.

There is an ongoing debate on the extent to which the writing of programs is an art, a craft or an engineering discipline.[1] Good programming is generally considered to be the measured application of all three, with the goal of producing an efficient and evolvable software solution (the criteria for "efficient" and "evolvable" vary considerably). The discipline differs from many other technical professions in that programmers generally do not need to be licensed or pass any standardized (or governmentally regulated) certification tests in order to call themselves "programmers" or even "software engineers".

Another ongoing debate is the extent to which the programming language used in writing computer programs affects the form that the final program takes. This debate is analogous to that surrounding the Sapir-Whorf hypothesis [2] in linguistics, that postulates that a particular language's nature influences the habitual thought of its speakers. Different language patterns yield different patterns of thought. This idea challenges the possibility of representing the world perfectly with language, because it acknowledges that the mechanisms of any language condition the thoughts of its speaker community.

Said another way, programming is the craft of transforming requirements into something that a computer can execute.

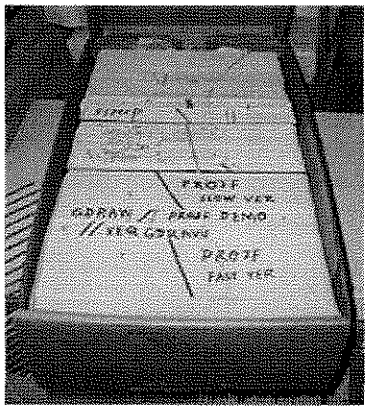
History of programming



Wired plug board for an IBM 402 Accounting Machine.

The concept of devices that operate following a pre-defined set of instructions traces back to Greek Mythology, notably Hephaestus and his mechanical servants[3]. The Antikythera mechanism was a calculator utilizing gears of various sizes and configuration to determine its operation. The earliest known programmable machines (machines whose behavior can be controlled and predicted with a set of instructions) were Al-Jazari's programmable Automata in 1206.[4] One of Al-Jazari's robots was originally a boat with four automatic musicians that floated on a lake to entertain guests at royal drinking parties. Programming this mechanism's behavior meant placing pegs and cams into a wooden drum at specific locations. These would then bump into little levers that operate a percussion instrument. The output of this device was a small drummer playing various rhythms and drum patterns.[5][6] Another sophisticated programmable machine by Al-Jazari was the castle clock, notable for its concept of variables which the operator could manipulate as necessary (i.e. the length of day and night). The Jacquard Loom, which Joseph Marie Jacquard developed in 1801, uses a series of pasteboard cards with holes punched in them. The hole pattern represented the pattern that the loom had to follow in weaving cloth. The loom could produce entirely different weaves using different sets of cards. Charles Babbage adopted the use of punched cards around 1830 to control his Analytical Engine. The synthesis of numerical calculation, predetermined operation and output, along with a way to organize and input instructions in a manner relatively easy for humans to conceive and produce, led to the modern development of computer programming.

Development of computer programming accelerated through the Industrial Revolution. The punch card innovation was later refined by Herman Hollerith who, in 1896 founded the *Tabulating Machine Company* (which became IBM). He invented the Hollerith punched card, the card reader, and the key punch machine. These inventions were the foundation of the modern information processing industry. The addition of a plug-board to his 1906 Type I Tabulator allowed it to do different jobs without having to be physically rebuilt. By the late 1940s there were a variety of plug-board programmable machines, called unit record equipment, to perform data processing tasks (card reading). Early computer programmers used plug-boards for the variety of complex calculations requested of the newly invented machines.



Data and instructions could be stored on external punch cards, which were kept in order and arranged in program decks.

The invention of the Von Neumann architecture allowed computer programs to be stored in computer memory. Early programs had to be painstakingly crafted using the instructions of the particular machine, often in binary notation. Every model of computer would be likely to need different instructions to do the same task. Later assembly languages were developed that let the programmer specify each instruction in a text format, entering abbreviations for each operation code instead of a number and specifying addresses in symbolic form (e.g. ADD X, TOTAL). In 1954 Fortran, the first higher level programming language, was invented. This allowed programmers to specify calculations by entering a formula directly (e.g. Y = X*2 + 5*X + 9). The program text, or *source*, was converted into machine instructions using a special program called a compiler. Many other languages were developed, including ones for commercial programming, such as COBOL. Programs were mostly still entered using punch cards or paper tape. (See computer programming in the punch card era). By the late 1960s, data storage devices and computer terminals became inexpensive enough so programs could be created by typing directly into the computers. Text editors were developed that allowed changes and corrections to be made much more easily than with punch cards.

As time has progressed, computers have made giant leaps in the area of processing power. This has brought about newer programming languages that are more abstracted from the underlying hardware. Although these more abstracted languages require additional overhead, in most cases the huge increase in speed of modern computers has brought about little performance decrease compared to earlier counterparts. The benefits of these more abstracted languages is that they allow both an easier learning curve for people less familiar with the older lower-level programming languages, and they also allow a more experienced programmer to develop simple applications quickly. Despite these benefits, large complicated programs, and programs that are more dependent on speed still require the faster and relatively lower-level languages with today's hardware. (The same concerns were raised about the original Fortran language.)