# Essay on Computer Programming

Article shared by : <="" div="" style="margin: 0px; padding: 0px; border: 0px; outline: 0px; font-size: 16px; vertical-align: bottom; background: transparent; max-width: 100%;">

ADVERTISEMENTS:

**The below mentioned essay provides a note on computer programming:- 1.** **Introduction to Computer Programming 2. Standard Computer Programmes 3. Debugging 4. Binary Code System 5. Decimal System 6. Distributed Data Processing (DDP) 7. Computer Generations 8. Ready-Made Software and Custom-Made Software.**

**Contents:**

1. Essay on Introduction to Computer Programming
2. Essay on Standard Computer Programmes
3. Essay on Debugging of Computer Programme
4. Essay on Binary Code System 5. Decimal System
5. Essay on Distributed Data Processing (DDP)
6. Essay on Computer Generations
7. Essay on Ready-Made Software and Custom-Made Software

**Essay # 1. Introduction to Computer Programming:**

ADVERTISEMENTS:

Programme is a sequence of instructions written in a proper language through which the computer can understand and solve the problem given to it. It is the method by which the whole computing process is directed and controlled. Preparing a programme for the computer is known as **“programming”.**

A programme should be recorded on a proper medium which the computer can process. Usually punched cards are used for this purpose. Each computer can understand one language which is known as **“machine language”.**

Machine language contains use of numeral codes and each computer has its own machine language. It is very difficult to write a programme in this language. To obliterate this difficulty, some other languages have been developed.

**These can be grouped into following two categories:**

A machine oriented language can only be used on a computer for which it has been designed. This language contains alphabetic codes, which are known as unmemoric codes. So, it is easier to remember unmemoric codes than numeric codes and it is easier to write a programme in this language.

A machine oriented language is oriented to a particular computer and not oriented to a particular problem. To avoid this difficulty, problem oriented languages have been developed. It is easier to write programmes in these languages. These are also known as high level languages.

These languages are also to be translated before execution. The programme used for translation in this case is known as ‘computer programme’. It is a standard programme written and supplied by the computer manufacturers for doing translation job.

It is a programme which translates programme in a language other than the machine code of a specific computer into instructions which a computer can obey. Because of this, language barrier between men and computer is broken down.

ADVERTISEMENTS:

**Some problem oriented languages are:**

FORTRAN (Formula Translation)

ALGOL (Algorithmic Language)

COBOL (Common Business Oriented Language)

ADVERTISEMENTS:

BASIC (Beginner’s All-purpose Symbolic Instruction Code)

Out of all FORTRAN is the most common programming language available on almost all the computer systems.

**Essay # 2. Standard Programmes:**

The computer manufacturers, as a matter of sales strategy, supply readymade programmes without additional charge which are common to many users. Such programmes are known as standard programmes because they are designed in a standard wav for applications and can be used by many users.

ADVERTISEMENTS:

Such programmes are becoming popular because of increased co-operation between the programme designers and the prospective users.

**Essay # 3. Debugging of Computer Programme:**

Very few programmes are accurate in the first instance. Usually there are errors in a programme. The removing of these errors (i.e., bugs) is known as debugging of a programme. There are two types of errors that a programmer has to deal with in a programme. These errors are known as syntax error and logical error.

Syntax error is the error that violates the grammar of the computer language in which the programme has been written. Logical error is that type of error which violates the logic of doing certain-thing. For example, an item which should have been added might have been deducted.

ADVERTISEMENTS:

**Essay # 4. Binary Code System:**

It is a method of expressing numerical value by numbering system based on two numbers. In this system, there are only two digits, 1 and 0. Zero is emergency digit and it is to be used when stock is exhausted. Use of zero is very frequent because the stock of one digit exhausts very quickly.

**Development of numbers in this system is as follows:**

Thus, in a binary code system the data is represented in terms of 0’s and l’s and this system is used to represent data on a computer.

ADVERTISEMENTS:

**Essay # 5. Decimal System:**

In this system there are ten digits— 1, 2, 3, 4, 5, 6, 7, 8, 9 and 0. The tenth digit zero is called emergency digit and is to be used when all other digits from 1 to 9 are exhausted. The zero digit is to be used in a systematic way i.e. first with first digit (10), then with second digit (20) and then with third digit (30) and so on and so forth. Thus, the base of decimal system is 10.

**Scanners:**

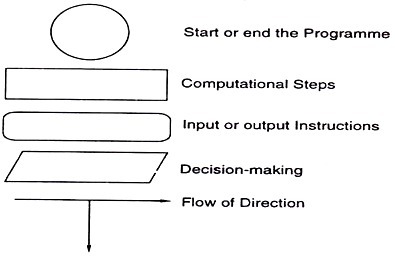
Scanners are devices which allow direct data entry into the computer without doing any manual data entry.

ADVERTISEMENTS:

**Flow Chart:**

A flow chart illustrates the sequence of operations to be performed to arrive at the solution of a problem. The operating instructions are placed in boxes which are connected by arrows to indicate the order of execution. These charts are an aid to writing programmes and are easier to understand at a glance than a narrative description. A flow charts also known as a flow diagram.

**While preparing flow charts, certain conventions have come into use as given below:**

**[](https://www.yourarticlelibrary.com/wp-content/uploads/2016/06/clip_image006-35.jpg)**

**A flow chart for the computation of Economic Order Quantity (i.e., EOQ) may be as follows:**

#### 

ADVERTISEMENTS:

**Essay # 6. Distributed Data Processing (DDP):**

DDP is a system in which the computing and other resources are decentralised at different places instead of being located at just one place. The places at which computers are installed are inter-connected.

In this system, there is some form of communication linkage between locations where the computer resources and users are situated. A manufacturing or trading or service organisation like banking can successfully make use of distributed data processing system.

**Internet:**

Any database situated in any corner of the world can be linked through the internet. Internet refers to network of different networks. The most important advantage of internet is the accessibility to information available anywhere in the world.

ADVERTISEMENTS:

Every organisation or individual using personal computer can create a web site in the internet and keep any sort of information on the web site. This information is accessible from anywhere in the world using computers.

Another advantage of internet is e-mail. E-mail is helpful in faster, better and cheaper communication across the world. Internet is also being used as a medium through which trade and aids to trade are carried out.

**Intranet:**

It is an information system which facilitates communication within the organisation, among widely dispersed departments, divisions and regional locations. It provides easily accessible information and reduces documentation cost. Information searching time is also reduced.

**Essay # 7. Computer Generations:**

Each computer generation corresponds to the introduction of radically advanced hardware technology with drastic changes in computing speeds, data storage capacities at low cost than previous generation computers and more versatile accompanied by commensurate changes in software characteristics.

ADVERTISEMENTS:

Most of the present day computers are known as fourth generation computers and the fifth generation computers with more advanced technology have also come in the market.

**Modem:**

Modem is an encoding/decoding device used in data transmission. It converts a digital computer signal into an analog telephone signal and converts an analog telephone signal into a digital computer signal in a data communication system.

**Stored Programme:**

The stored programme is under the command of the control unit of the central processing unit (CPU) and allows the computer to process data automatically without continual human intervention at different stages of processing.

**Essay # 8. Ready-Made Software and Custom-Made Software:**

ADVERTISEMENTS:

Ready-made software are standard programmes which are common to many users. Such programmers are developed by the computer companies for the benefit of many users. The cost of ready-made software is less and time for implementation is less. Modification in such a software is costly and sometimes not possible.

On the other hand custom-made software is developed according to the specific requirements of a user. Good amount of time is required for the development of a custom-made software and it is very costly as compared to a ready-made software. Incorporation of modification is easy in custom-made software and hardware set up generally need not to be changed.

**Parity Bit:**

Parity bit is one of the most effective hardware controls which make possible timely detection of errors. During transmission/transfer of individual characters on input/output medium, it is possible that a malfunctioning caused by dirt, humidity etc. results in loss or gain of an extra bit in a particular bit string corrupting data thereby.

The use of parity bit in such cases is helpful in timely detection of such type of errors.

**E-Commerce:**

E-commerce is the abbreviated term for electronic commerce. It is a way of conducting business electronically with the help of the internet technology. Conducting business on the web is E-commerce. It is helpful in opening new markets on global scale. Any two business houses using the internet can interact with one other.

**Computer Checks:**

The objective of installing a computer is to have high level of accuracy. But unfortunately it exercises no judgment as is possible in case of a manual system. Therefore, it cannot reject data which are wrongly fed.

Keeping in view this drawback of a computer of not noticing immediately certain types of errors as are noticed in a manual system by persons dealing with data, a play card carrying the message GIGO is hung at computer installations so as to caution the operating staff. GIGO means ‘Garbage in Garbage out.’

The idea of this phrase is that if the information fed (i.e. input) is defective any amount of processing cannot make it all right. Therefore, to restrict errors certain hardware and software checks are applied as given below:

**Parity Check:**

It is a means of checking that information is not lost.

**Echo Check:**

By this check we mean that whatever is written on magnetic tape is also simultaneously read for cross checking.

**Limit Check:**

Every field of data like hours worked by an employee has the minimum and maximum limits. The programme should ensure that the data are within the predefined limits.

**Compatibility Check:**

In this check, a formula is worked in different ways to ensure that the same results are obtained.

**Sequence Check:**

In this type of check, programming is done in such a way that if the sequence is missing, the computer gives signal for the missing transaction. In addition to these checks, if the user department using the mechanised data comes across any error should report immediately so that the programmes may be suitably amended.

**Parity Check Bit:**

It is an extra bit used to store characters in computers in order to prevent and locate possible coding error.

**Computer programming** or **coding** is the composition of sequences of instructions, called [programs](https://en.wikipedia.org/wiki/Computer_program), that [computers](https://en.wikipedia.org/wiki/Computer) can follow to perform tasks.[[1]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-2) It involves designing and implementing [algorithms](https://en.wikipedia.org/wiki/Algorithm), step-by-step specifications of procedures, by writing [code](https://en.wikipedia.org/wiki/Source_code) in one or more [programming languages](https://en.wikipedia.org/wiki/Programming_language). Programmers typically use [high-level programming languages](https://en.wikipedia.org/wiki/High-level_programming_language) that are more easily intelligible to humans than [machine code](https://en.wikipedia.org/wiki/Machine_code), which is directly executed by the [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit). Proficient programming usually requires expertise in several different subjects, including knowledge of the [application domain](https://en.wikipedia.org/wiki/Domain_(software_engineering)), details of programming languages and generic code [libraries](https://en.wikipedia.org/wiki/Library_(computing)), specialized algorithms, and formal [logic](https://en.wikipedia.org/wiki/Logic).

Auxiliary tasks accompanying and related to programming include [analyzing requirements](https://en.wikipedia.org/wiki/Requirements_analysis), [testing](https://en.wikipedia.org/wiki/Software_testing), [debugging](https://en.wikipedia.org/wiki/Debugging) (investigating and fixing problems), implementation of [build systems](https://en.wikipedia.org/wiki/Build_automation), and management of derived [artifacts](https://en.wikipedia.org/wiki/Artifact_(software_development)), such as programs' [machine code](https://en.wikipedia.org/wiki/Machine_code). While these are sometimes considered programming, often the term [*software development*](https://en.wikipedia.org/wiki/Software_development) is used for this larger overall process – with the terms *programming*, *implementation*, and *coding* reserved for the writing and editing of code per se. Sometimes software development is known as [*software engineering*](https://en.wikipedia.org/wiki/Software_engineering), especially when it employs [formal methods](https://en.wikipedia.org/wiki/Formal_methods) or follows an [engineering design process](https://en.wikipedia.org/wiki/Engineering_design_process).

## History[

## [Ada Lovelace](https://en.wikipedia.org/wiki/Ada_Lovelace), whose notes added to the end of [Luigi Menabrea](https://en.wikipedia.org/wiki/Luigi_Menabrea)'s paper included the first [algorithm](https://en.wikipedia.org/wiki/Algorithm) designed for processing by [Charles Babbage](https://en.wikipedia.org/wiki/Charles_Babbage)'s [Analytical Engine](https://en.wikipedia.org/wiki/Analytical_Engine). She is often recognized as history's first computer programmer.

*See also:*[*Computer program § History*](https://en.wikipedia.org/wiki/Computer_program#History)*,*[*Programmer § History*](https://en.wikipedia.org/wiki/Programmer#History)*, and*[*History of programming languages*](https://en.wikipedia.org/wiki/History_of_programming_languages)

[Programmable devices](https://en.wikipedia.org/wiki/Program_(machine)) have existed for centuries. As early as the 9th century, a programmable [music sequencer](https://en.wikipedia.org/wiki/Music_sequencer) was invented by the Persian [Banu Musa](https://en.wikipedia.org/wiki/Banu_Musa) brothers, who described an automated mechanical [flute](https://en.wikipedia.org/wiki/Flute) player in the [*Book of Ingenious Devices*](https://en.wikipedia.org/wiki/Book_of_Ingenious_Devices).[[3]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-Koetsier-3)[[4]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-4) In 1206, the Arab engineer [Al-Jazari](https://en.wikipedia.org/wiki/Al-Jazari) invented a programmable [drum machine](https://en.wikipedia.org/wiki/Drum_machine) where a musical mechanical [automaton](https://en.wikipedia.org/wiki/Automaton) could be made to play different rhythms and drum patterns, via pegs and [cams](https://en.wikipedia.org/wiki/Cam_(mechanism)).[[5]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-Sharkey-6) In 1801, the [Jacquard loom](https://en.wikipedia.org/wiki/Jacquard_loom) could produce entirely different weaves by changing the "program" – a series of [pasteboard](https://en.wikipedia.org/wiki/Card_stock) cards with holes punched in them.

[Code-breaking](https://en.wikipedia.org/wiki/Code-breaking) algorithms have also existed for centuries. In the 9th century, the [Arab mathematician](https://en.wikipedia.org/wiki/Mathematics_in_medieval_Islam) [Al-Kindi](https://en.wikipedia.org/wiki/Al-Kindi) described a [cryptographic](https://en.wikipedia.org/wiki/Cryptographic) algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. He gave the first description of [cryptanalysis](https://en.wikipedia.org/wiki/Cryptanalysis) by [frequency analysis](https://en.wikipedia.org/wiki/Frequency_analysis), the earliest code-breaking algorithm.[[7]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-7)

The first [computer program](https://en.wikipedia.org/wiki/Computer_program) is generally dated to 1843 when mathematician [Ada Lovelace](https://en.wikipedia.org/wiki/Ada_Lovelace) published an [algorithm](https://en.wikipedia.org/wiki/Algorithm) to calculate a sequence of [Bernoulli numbers](https://en.wikipedia.org/wiki/Bernoulli_numbers), intended to be carried out by [Charles Babbage](https://en.wikipedia.org/wiki/Charles_Babbage)'s [Analytical Engine](https://en.wikipedia.org/wiki/Analytical_Engine).[[8]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-IEEE-8)

Data and instructions were once stored on external [punched cards](https://en.wikipedia.org/wiki/Punched_card), which were kept in order and arranged in program decks.

In the 1880s, [Herman Hollerith](https://en.wikipedia.org/wiki/Herman_Hollerith) invented the concept of storing *data* in machine-readable form.[[9]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-9) Later a [control panel](https://en.wikipedia.org/wiki/Plugboard) (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, [unit record equipment](https://en.wikipedia.org/wiki/Unit_record_equipment) such as the [IBM 602](https://en.wikipedia.org/wiki/IBM_602) and [IBM 604](https://en.wikipedia.org/wiki/IBM_604), were programmed by control panels in a similar way, as were the first [electronic computers](https://en.wikipedia.org/wiki/Electronic_computer). However, with the concept of the [stored-program computer](https://en.wikipedia.org/wiki/Stored-program_computer) introduced in 1949, both programs and data were stored and manipulated in the same way in [computer memory](https://en.wikipedia.org/wiki/Computer_memory).[[10]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-10)

### Machine language[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=2)]

[Machine code](https://en.wikipedia.org/wiki/Machine_code) was the language of early programs, written in the [instruction set](https://en.wikipedia.org/wiki/Instruction_set_architecture) of the particular machine, often in [binary](https://en.wikipedia.org/wiki/Binary_numeral_system) notation. [Assembly languages](https://en.wikipedia.org/wiki/Assembly_language) were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. However, because an assembly language is little more than a different notation for a machine language, two machines with [different instruction sets](https://en.wikipedia.org/wiki/Comparison_of_instruction_set_architectures) also have different assembly languages.

Wired [control panel](https://en.wikipedia.org/wiki/Plugboard) for an [IBM 402 Accounting Machine](https://en.wikipedia.org/wiki/IBM_402_Accounting_Machine). Wires connect pulse streams from the card reader to counters and other internal logic and ultimately to the printer.

### Compiler languages[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=3)]

*See also:*[*Compiler*](https://en.wikipedia.org/wiki/Compiler)

[High-level languages](https://en.wikipedia.org/wiki/High-level_language) made the process of developing a program simpler and more understandable, and less bound to the underlying [hardware](https://en.wikipedia.org/wiki/Computer_hardware). The first compiler related tool, the [A-0 System](https://en.wikipedia.org/wiki/A-0_System), was developed in 1952[[11]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-11) by [Grace Hopper](https://en.wikipedia.org/wiki/Grace_Hopper), who also coined the term 'compiler'.[[12]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-wikles1968-12)[[13]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-computerhistory.org-13) [FORTRAN](https://en.wikipedia.org/wiki/FORTRAN), the first widely used high-level language to have a functional implementation, came out in 1957,[[14]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-bergstein-14) and many other languages were soon developed—in particular, [COBOL](https://en.wikipedia.org/wiki/COBOL) aimed at commercial data processing, and [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) for computer research.

These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of [abstracting](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) the code, making it easy to target varying machine instruction sets via compilation declarations and [heuristics](https://en.wikipedia.org/wiki/Heuristic_(computer_science)). Compilers harnessed the power of computers to make programming easier[[14]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-bergstein-14) by allowing programmers to specify calculations by entering a formula using [infix notation](https://en.wikipedia.org/wiki/Infix_notation).

### Source code entry[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=4)]

*See also:*[*Computer programming in the punched card era*](https://en.wikipedia.org/wiki/Computer_programming_in_the_punched_card_era)

Programs were mostly entered using punched cards or [paper tape](https://en.wikipedia.org/wiki/Paper_tape). By the late 1960s, [data storage devices](https://en.wikipedia.org/wiki/Data_storage_device) and [computer terminals](https://en.wikipedia.org/wiki/Computer_terminal) became inexpensive enough that programs could be created by typing directly into the computers. [Text editors](https://en.wikipedia.org/wiki/Text_editor) were also developed that allowed changes and corrections to be made much more easily than with [punched cards](https://en.wikipedia.org/wiki/Punched_card_sorter).

## Modern programming[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=5)]

### Quality requirements[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=6)]

*Main article:*[*Software quality*](https://en.wikipedia.org/wiki/Software_quality)

Whatever the approach to development may be, the final program must satisfy some fundamental properties. The following properties are among the most important:[[15]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-15) [[16]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-16)

* [Reliability](https://en.wikipedia.org/wiki/Reliability_engineering#Software_reliability): how often the results of a program are correct. This depends on conceptual correctness of algorithms and minimization of programming mistakes, such as mistakes in resource management (e.g., [buffer overflows](https://en.wikipedia.org/wiki/Buffer_overflow) and [race conditions](https://en.wikipedia.org/wiki/Race_condition)) and logic errors (such as division by zero or [off-by-one errors](https://en.wikipedia.org/wiki/Off-by-one_error)).
* [Robustness](https://en.wikipedia.org/wiki/Robustness_(computer_science)): how well a program anticipates problems due to errors (not bugs). This includes situations such as incorrect, inappropriate or corrupt data, unavailability of needed resources such as memory, operating system services, and network connections, user error, and unexpected power outages.
* [Usability](https://en.wikipedia.org/wiki/Usability): the [ergonomics](https://en.wikipedia.org/wiki/Ergonomics) of a program: the ease with which a person can use the program for its intended purpose or in some cases even unanticipated purposes. Such issues can make or break its success even regardless of other issues. This involves a wide range of textual, graphical, and sometimes hardware elements that improve the clarity, intuitiveness, cohesiveness, and completeness of a program's user interface.
* [Portability](https://en.wikipedia.org/wiki/Software_portability): the range of [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware) and [operating system](https://en.wikipedia.org/wiki/Operating_system) platforms on which the source code of a program can be [compiled](https://en.wikipedia.org/wiki/Compiled)/[interpreted](https://en.wikipedia.org/wiki/Interpreter_(computing)) and run. This depends on differences in the programming facilities provided by the different platforms, including hardware and operating system resources, expected behavior of the hardware and operating system, and availability of platform-specific compilers (and sometimes libraries) for the language of the source code.
* [Maintainability](https://en.wikipedia.org/wiki/Maintainability): the ease with which a program can be modified by its present or future developers in order to make improvements or to customize, fix [bugs](https://en.wikipedia.org/wiki/Software_bug) and [security holes](https://en.wikipedia.org/wiki/Vulnerability_(computing)), or adapt it to new environments. Good practices[[17]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-17) during initial development make the difference in this regard. This quality may not be directly apparent to the end user but it can significantly affect the fate of a program over the long term.
* [Efficiency](https://en.wikipedia.org/wiki/Algorithmic_efficiency)/[performance](https://en.wikipedia.org/wiki/Performance_engineering): Measure of system resources a program consumes (processor time, memory space, slow devices such as disks, network bandwidth and to some extent even user interaction): the less, the better. This also includes careful management of resources, for example cleaning up [temporary files](https://en.wikipedia.org/wiki/Temporary_file) and eliminating [memory leaks](https://en.wikipedia.org/wiki/Memory_leak). This is often discussed under the shadow of a chosen programming language. Although the language certainly affects performance, even slower languages, such as [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), can execute programs instantly from a human perspective. Speed, resource usage, and performance are important for programs that [bottleneck](https://en.wikipedia.org/wiki/Bottleneck_(software)) the system, but efficient use of programmer time is also important and is related to cost: more hardware may be cheaper.

### Readability of source code[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=7)]

In computer programming, [readability](https://en.wikipedia.org/wiki/Readability) refers to the ease with which a human reader can comprehend the purpose, [control flow](https://en.wikipedia.org/wiki/Control_flow), and operation of source code. It affects the aspects of quality above, including portability, usability and most importantly maintainability.

Readability is important because programmers spend the majority of their time reading, trying to understand, reusing, and modifying existing source code, rather than writing new source code. Unreadable code often leads to bugs, inefficiencies, and [duplicated code](https://en.wikipedia.org/wiki/Code_duplication). A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.[[18]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-18)

Following a consistent [programming style](https://en.wikipedia.org/wiki/Programming_style) often helps readability. However, readability is more than just programming style. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.[[19]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-19) Some of these factors include:

* Different [indent styles](https://en.wikipedia.org/wiki/Indent_style) (whitespace)
* [Comments](https://en.wikipedia.org/wiki/Comment_(computer_programming))
* [Decomposition](https://en.wikipedia.org/wiki/Decomposition_(computer_science))
* [Naming conventions](https://en.wikipedia.org/wiki/Naming_conventions_(programming)) for objects (such as variables, classes, functions, procedures, etc.)

The [presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content) aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the [source code editor](https://en.wikipedia.org/wiki/Source_code_editor), but the content aspects reflect the programmer's talent and skills.

Various [visual programming languages](https://en.wikipedia.org/wiki/Visual_programming_language) have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. [Integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDEs) aim to integrate all such help. Techniques like [Code refactoring](https://en.wikipedia.org/wiki/Code_refactoring) can enhance readability.

### Algorithmic complexity[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=8)]

The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. For this purpose, algorithms are classified into *orders* using so-called [Big O notation](https://en.wikipedia.org/wiki/Big_O_notation), which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.

### Methodologies[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=9)]

The first step in most formal software development processes is [requirements analysis](https://en.wikipedia.org/wiki/Requirements_analysis), followed by testing to determine value modeling, implementation, and failure elimination (debugging). There exist a lot of different approaches for each of those tasks. One approach popular for requirements analysis is [Use Case](https://en.wikipedia.org/wiki/Use_Case) analysis. Many programmers use forms of [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. There are many approaches to the Software development process.

Popular modeling techniques include Object-Oriented Analysis and Design ([OOAD](https://en.wikipedia.org/wiki/OOAD)) and Model-Driven Architecture ([MDA](https://en.wikipedia.org/wiki/Model-Driven_Architecture)). The Unified Modeling Language ([UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language)) is a notation used for both the OOAD and MDA.

A similar technique used for database design is Entity-Relationship Modeling ([ER Modeling](https://en.wikipedia.org/wiki/Entity-Relationship_Model)).

Implementation techniques include imperative languages ([object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) or [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), [functional languages](https://en.wikipedia.org/wiki/Functional_programming), and [logic languages](https://en.wikipedia.org/wiki/Logic_programming).

### Measuring language usage[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=10)]

It is very difficult to determine what are the most popular modern programming languages. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language,[[20]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-20) the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).

Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. For example, [COBOL](https://en.wikipedia.org/wiki/COBOL) is still strong in corporate data centers[[21]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-21) often on large [mainframe computers](https://en.wikipedia.org/wiki/Mainframe_computer), [Fortran](https://en.wikipedia.org/wiki/Fortran) in engineering applications, [scripting languages](https://en.wikipedia.org/wiki/Scripting_language) in [Web](https://en.wikipedia.org/wiki/World_Wide_Web) development, and [C](https://en.wikipedia.org/wiki/C_(programming_language)) in [embedded software](https://en.wikipedia.org/wiki/Embedded_software). Many applications use a mix of several languages in their construction and use. New languages are generally designed around the syntax of a prior language with new functionality added, (for example [C++](https://en.wikipedia.org/wiki/C%2B%2B) adds object-orientation to C, and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) adds memory management and [bytecode](https://en.wikipedia.org/wiki/Bytecode) to C++, but as a result, loses efficiency and the ability for low-level manipulation).

### Debugging[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=11)]

*Main article:*[*Debugging*](https://en.wikipedia.org/wiki/Debugging)

[](https://en.wikipedia.org/wiki/File:First_Computer_Bug,_1945.jpg)The first known actual bug causing a problem in a computer was a moth, trapped inside a Harvard mainframe, recorded in a log book entry dated September 9, 1947.[[22]](https://en.wikipedia.org/wiki/Computer_programming#cite_note-22) "Bug" was already a common term for a software defect when this insect was found.

Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Use of a [static code analysis](https://en.wikipedia.org/wiki/Static_code_analysis) tool can help detect some possible problems. Normally the first step in debugging is to attempt to reproduce the problem. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Also, specific user environment and usage history can make it difficult to reproduce the problem.

After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. For example, when a bug in a compiler can make it crash when [parsing](https://en.wikipedia.org/wiki/Parsing) some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. Scripting and [breakpointing](https://en.wikipedia.org/wiki/Breakpoint) is also part of this process.

Debugging is often done with [IDEs](https://en.wikipedia.org/wiki/Integrated_development_environment). Standalone debuggers like [GDB](https://en.wikipedia.org/wiki/GDB) are also used, and these often provide less of a visual environment, usually using a [command line](https://en.wikipedia.org/wiki/Command_line). Some text editors such as [Emacs](https://en.wikipedia.org/wiki/Emacs) allow GDB to be invoked through them, to provide a visual environment.

## Programming languages[[edit](https://en.wikipedia.org/w/index.php?title=Computer_programming&action=edit&section=12)]

*Main articles:*[*Programming language*](https://en.wikipedia.org/wiki/Programming_language)*and*[*List of programming languages*](https://en.wikipedia.org/wiki/List_of_programming_languages)

*See also:*[*Computer program § Languages*](https://en.wikipedia.org/wiki/Computer_program#Languages)

Different programming languages support different styles of programming (called [*programming paradigms*](https://en.wikipedia.org/wiki/Programming_paradigm)). The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Ideally, the programming language best suited for the task at hand will be selected. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. It is usually easier to code in "high-level" languages than in "low-level" ones. Programming languages are essential for software development. They are the building blocks for all software, from the simplest applications to the most sophisticated ones.

[Allen Downey](https://en.wikipedia.org/wiki/Allen_Downey), in his book *How To Think Like A Computer Scientist*, writes:

The details look different in different languages, but a few basic instructions appear in just about every language:

* Input: Gather data from the keyboard, a file, or some other device.
* Output: Display data on the screen or send data to a file or other device.
* Arithmetic: Perform basic arithmetical operations like addition and multiplication.
* Conditional Execution: Check for certain conditions and execute the appropriate sequence of statements.
* Repetition: Perform some action repeatedly, usually with some variation.

Many computer languages provide a mechanism to call functions provided by [shared libraries](https://en.wikipedia.org/wiki/Shared_library). Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing [arguments](https://en.wikipedia.org/wiki/Argument_(computer_science))), then these functions may be written in any other language.

The career development process can be complex and overwhelming—whether you’re just starting in your field or looking to make a change.

To simplify the process, we’ve broken it down into five key steps: self-assessment, goal setting, action planning, implementation, and refinement. By following these steps, you can develop a road map for achieving your career goals.

Ready to take your career to the next level? This comprehensive guide on career development has everything you need to know to jump-start your professional journey.

* [What is career development?](https://www.upwork.com/resources/what-is-career-development#career-development)
* [How career development works](https://www.upwork.com/resources/what-is-career-development#how-it-works)
* [Types of career development](https://www.upwork.com/resources/what-is-career-development#types)
* [How to develop your career](https://www.upwork.com/resources/what-is-career-development#how-to-development)
* [Benefits and challenges of career development](https://www.upwork.com/resources/what-is-career-development#benefits-and-challenges)
* [Tips for effective career development](https://www.upwork.com/resources/what-is-career-development#tips)

## **What is career development?**

Career development is the proactive, lifelong process of finding your footing and advancing your career path. It’s an intentional approach to creating a meaningful career that includes setting long-term goals, exploring professional development opportunities, and gaining new work experience.

In today’s job market, career development is increasingly important. With changes in technology and the global economy, professionals must adapt to stay competitive. A strategic [**career development plan**](https://www.upwork.com/resources/career-development-plan)helps make this possible.

## **How career development works**

Career development can take different forms, but people generally take two paths: through an organization or independently.

Professionals who develop their careers through an organization tend to have more resources available. Their company may offer development programs, as well as a support network of employees.

Independent professionals often must look for career development opportunities on their own. They might use online resources like social media and online classes, attend conferences or networking events, meet with a career counselor, or join professional organizations and read trade publications.

Because independent professionals don’t have the same resources as those who work within organizations, they may need to be more proactive in their career growth.

## **Types of career development**

Two types of career development are:

* **Formal career development.**This includes short-term training programs, education, certifications, workshops, or seminars that can help build skill sets for a particular job or industry.
* **Informal career development.**This includes mentorship opportunities, networking events, online courses, internships, and volunteering experiences.

Explore both formal and informal methods to ensure you have updated information relevant to your career journey.

### **Career development examples**

These career development scenarios can give you a glimpse into different career options and how they can develop into your dream job:

**1. Creative careers**

* A graphic designer who starts in a small agency, gains competencies, and opens their own studio
* A filmmaker who starts as a camera operator, networks with the production crew, and works their way to become a film director or producer

**2. Technical careers**

* A software developer who begins as a junior developer, earns certifications, and becomes a lead developer
* A network administrator who starts as a support technician, gains experience, mentors under the lead, and becomes a network architect

**3. Business careers**

* A sales representative who begins as an entry-level representative, learns the ropes, takes business classes, and becomes a sales manager
* A Human Resources (HR) professional who starts as a recruiter, gains experience, works with a business coach, and becomes a Chief HR Officer (CHRO)

These are just a few examples of the many career paths available.

## **How to develop your career**

Following a series of steps, you can create a road map for professional growth and achieve your career goals.

**Steps in the career development process:**

1. [Self-assessment](https://www.upwork.com/resources/what-is-career-development#self-assesment)
2. [Goal setting](https://www.upwork.com/resources/what-is-career-development#goal-setting)
3. [Action planning](https://www.upwork.com/resources/what-is-career-development#action-planning)
4. [Implementation](https://www.upwork.com/resources/what-is-career-development#implementation)
5. [Refinement](https://www.upwork.com/resources/what-is-career-development#refinement)

### **1. Self-assessment**

A self-assessment evaluates your strengths, weaknesses, skills, personality type, interests, and values to help you understand your fit with various career paths.

Self-assessment examples include:

* **Skills assessment.**This type of self-assessment helps identify your strengths and determine skills gaps where you need development. Consider taking a [leadership self-assessment](https://www.upwork.com/resources/leadership-self-assessment-examples) for a specific evaluation of your management skills.
* **Interest inventory.**This assessment helps you understand your passions to help you explore potential career paths.
* **Values assessment.**This identifies your values and priorities, which can affect your career satisfaction.

By completing a self-assessment, you can better understand your personal career needs and successfully guide your career development.

### **2. Goal setting**

Using your self-assessment to [**set career goals**](https://www.upwork.com/resources/career-development-goals) is a critical step in career development. Explore these career goal examples based on a self-assessment:

* **Skills development.** If you find a gap in your skills, you might set a goal to get professional training or enroll in college courses.
* **Career advancement.**You may set a goal to move up the career ladder based on your interests and values.
* **Career change.** If your current career doesn’t align with your interests and values, you may set a goal to transition to a different field.

Setting achievable goals allows you to focus and progress toward your objectives.

### **3. Action planning**

Action planning involves breaking down your career aspirations into manageable, achievable tasks and developing a timeline for completing them.

Three examples of what your action plan can look like:

* **Skills development.** If you want to acquire a new skill, your action plan may include researching training programs over the next week, enrolling in courses by the end of the month, and implementing the new skill in your career within the next quarter.
* **Career advancement.**Your plan may include networking at the next business event, volunteering for a specific upcoming leadership opportunity, hiring a business coach next month, or seeking feedback from leadership once a week over the next quarter.
* **Career change.**Your plan could include taking a month to research careers that align with your values, enrolling in certification courses or classes to gain skills in the new field over the next several months, networking with businesses, applying for positions in the new field over the next six to nine months, and transitioning to the new career field by next year.

By breaking down your aspirations into small tasks, you can stay focused, track milestones, and increase your chances of success.

### **4. Implementation**

Implementing your action plan is next. Discover three ways you can do this:

* **Schedule time.**Set aside time to focus on your goals. You might use this time to develop new skills, network, or research career paths.
* **Track progress.**Record your progress in a journal, spreadsheet, or to-do list. If you need to catch up on the timeline of your initial action plan, strategize ways to invest more time and effort into your development to get back on track.
* **Celebrate wins.** Celebrate your accomplishments, whether landing a new job, completing a certification, or making connections.

Taking action and regularly monitoring your progress can help you make meaningful strides toward your career goals.

### **5. Refinement**

Refining the career development process to go more smoothly over time involves regularly reassessing your goals, plans, and progress. You can do this in a few ways:

* **Regularly reflect.** Set aside time each month, quarter, or year to reflect on your career development process. Review your career goals, assess your progress, and identify areas for improvement.
* **Seek feedback.** Seek feedback from others with experience in your desired career. This feedback can provide valuable insights and help you refine your action plan.
* **Stay adaptable.**Be open to changing your goals and strategies. This may involve changing your timeline, pivoting to a new career path, or acquiring new skills.

This approach is designed to help you continuously grow and reach your career aspirations.

[Join the world's work marketplace](https://www.upwork.com/nx/signup/?dest=home.)

[Get Started](https://www.upwork.com/nx/signup/?dest=home.)

‍

## **Benefits and challenges of career development**

Understanding the benefits and challenges of career development can make a difference in how successful you are in advancing your career.

For instance, if you know one benefit of career development is the chance to learn new skills, you can try to find training opportunities. And if you know about a potential challenge, like limited resources, you can find alternative ways to get support.

### **Benefits**

Career development can bring you numerous benefits, including:

* **Increased satisfaction.** Career development can help you find fulfillment by allowing you to pursue your passions and strengths.
* **Higher earning potential.** Developing new skills and gaining new experiences can make you more valuable to employers.
* **Improved performance.**Career development can help you stay current in your field and make you more effective.
* **Greater opportunities for advancement:**Career development can help you move up the ladder or explore new career paths.
* **Enhanced confidence.**Pursuing career development can boost your self-esteem, making you feel more capable.
* **Improved work-life balance.**Career development can help you create a more fulfilling career.

By investing in yourself and your career, you can achieve greater success overall.

### **Challenges**

While career development can bring numerous benefits, it can also present challenges, including:

* **Limited opportunities.**Career development can be expensive and require resources that may not be immediately available.
* **Lack of support.**Employers may not provide support for career development, such as time off or compensation.
* **Difficulty in pursuing goals.**It can be challenging to identify career goals, determine how to reach them, or find the motivation to pursue them.
* **Balancing career and personal life.**Pursuing career development can be time-consuming and demanding.
* **Staying current.** Keeping up with the latest trends in your field can be challenging, requiring continuous education.
* **Overcoming self-doubt.**Pursuing career development can be intimidating, and many people struggle with their own lack of confidence.

When you acknowledge these potential challenges, you can be better prepared to overcome them and reach your goals.

## **Tips for effective career development**

For successful career development, you’ll want to follow some best practices:

* **Improve your communication skills.**Communication is crucial in the workplace. Consider taking courses or workshops to enhance your skills.
* **Invest in your education.** Stay current by taking courses, attending conferences, or pursuing a degree or certification.
* **Be intentional.** Set specific, achievable career goals, and be intentional about pursuing them.
* **Build a strong network.** Building relationships with others in your field can provide opportunities for advancement.
* **Take the initiative.**Seek new opportunities and get involved in projects that interest you.
* **Find the right mentor.**Look for someone experienced in your field with the qualities you want to emulate.
* **Explore what interests you.**Experiment with different roles, industries, and projects to find what interests you.

These best practices can help set you up for success as you pursue career development.