

It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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, 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). There exist a lot of different approaches for each of those tasks. Normally the first step in debugging is to attempt to reproduce the problem. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. For example, when a bug in a compiler can make it crash when 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. Techniques like Code refactoring can enhance readability. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. There are many approaches to the Software development process. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Techniques like Code refactoring can enhance readability. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Normally the first step in debugging is to attempt to reproduce the problem. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). They are the building blocks for all software, from the simplest applications to the most sophisticated ones. It affects the aspects of quality above, including portability, usability and most importantly maintainability. There are many approaches to the Software development process. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.