Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
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.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Also, those involved with software development may at times engage in reverse engineering, which is the practice of seeking to understand an existing program so as to re-implement its function in some way.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
One approach popular for requirements analysis is Use Case analysis.  
This is interpreted into machine code.  
 It is very difficult to determine what are the most popular modern programming languages.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.