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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. 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). 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. It is very difficult to determine what are the most popular modern programming languages. While these are sometimes considered programming, often the term 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. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). Normally the first step in debugging is to attempt to reproduce the problem. 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. Techniques like Code refactoring can enhance readability. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Programs were mostly entered using punched cards or paper tape. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. Programmable devices have existed for centuries. Also, specific user environment and usage history can make it difficult to reproduce the problem. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. It is very difficult to determine what are the most popular modern programming languages. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.