**UNIT 15**

**Computer Programming**

Computer programming is the process of writing, testing, [debugging](http://en.wikipedia.org/wiki/Debugging)/[troubleshooting](http://en.wikipedia.org/wiki/Troubleshooting), and maintaining the [source code](http://en.wikipedia.org/wiki/Source_code) of [computer programs](http://en.wikipedia.org/wiki/Computer_program). This source code is written in a [programming language](http://en.wikipedia.org/wiki/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 behaviour (customization). The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized [algorithms](http://en.wikipedia.org/wiki/Algorithm) and [formal logic](http://en.wikipedia.org/wiki/Logic).

**Overview**

There is an ongoing debate on the extent to which the writing of programs is an [art](http://en.wikipedia.org/wiki/Art), a [craft](http://en.wikipedia.org/wiki/Craft) or an [engineering](http://en.wikipedia.org/wiki/Engineering) discipline. In general, good programming is 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](http://en.wikipedia.org/wiki/Programmer), in general, 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’.

Said another way, programming is the craft of transforming [requirements](http://en.wikipedia.org/wiki/Requirement) into something that a [computer](http://en.wikipedia.org/wiki/Computer) can execute.

**Quality requirements**

Whatever the approach to software development may be, the final program must satisfy some fundamental properties. The following properties are among the most relevant ones:

- [efficiency](http://en.wikipedia.org/wiki/Algorithmic_efficiency)/[performance](http://en.wikipedia.org/wiki/Performance_engineering): the amount 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 correct disposal of some resources, such as cleaning up [temporary files](http://en.wikipedia.org/wiki/Temporary_file) and lack of [memory leaks](http://en.wikipedia.org/wiki/Memory_leak);

- [reliability](http://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](http://en.wikipedia.org/wiki/Buffer_overflow) and [race conditions](http://en.wikipedia.org/wiki/Race_condition)) and logic errors (such as division by zero);

- [robustness](http://en.wikipedia.org/wiki/Robustness): how well a program anticipates problems not due to programmer’s error. This includes situations such as incorrect, inappropriate or corrupt data, unavailability of needed resources such as memory, operating system services and network connections, and user error;

- [usability](http://en.wikipedia.org/wiki/Usability): the [ergonomics](http://en.wikipedia.org/wiki/Ergonomics) of a program: the ease with which a person can use the program for its intended purposes, 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](http://en.wikipedia.org/wiki/Software_portability): the range of [computer hardware](http://en.wikipedia.org/wiki/Computer_hardware) and [operating system](http://en.wikipedia.org/wiki/Operating_system) platforms on which the [source code](http://en.wikipedia.org/wiki/Source_code) of a program can be [compiled](http://en.wikipedia.org/wiki/Compiler)/[interpreted](http://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 behaviour of the hardware and operating system, and availability of platform specific compilers (and sometimes libraries) for the language of the source code;

- [maintainability](http://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 customizations, fix [bugs](http://en.wikipedia.org/wiki/Software_bug) and [security holes](http://en.wikipedia.org/wiki/Vulnerability_(computing)), or adapt it to new environments. Good practices 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.

**Measuring language usage**

It is very difficult to determine what are the most popular of modern programming languages. Some languages are very popular for particular kinds of applications (e.g., [COBOL](http://en.wikipedia.org/wiki/COBOL) is still strong in the corporate data center, often on large [mainframes](http://en.wikipedia.org/wiki/Mainframe_computer), [FORTRAN](http://en.wikipedia.org/wiki/Fortran_(programming_language)) in engineering applications, [scripting languages](http://en.wikipedia.org/wiki/Scripting_language) in web development, and [C](http://en.wikipedia.org/wiki/C_(programming_language)) in [embedded applications](http://en.wikipedia.org/wiki/Embedded_software)), while some languages are regularly used to write many different kinds of applications.

**Debugging**

[Debugging](http://en.wikipedia.org/wiki/Debugging) is a very important task in the software development process, because an incorrect 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](http://en.wikipedia.org/wiki/Compiler) to perform as much checking as other languages. Use of a [static analysis](http://en.wikipedia.org/wiki/Static_analysis) tool can help detect some possible problems.

Debugging is often done with [IDEs](http://en.wikipedia.org/wiki/Integrated_development_environment) (Integrated Development Environments) like [Visual Studio](http://en.wikipedia.org/wiki/Visual_Studio), [NetBeans](http://en.wikipedia.org/wiki/NetBeans), and [Eclipse](http://en.wikipedia.org/wiki/Eclipse_(software)). Standalone debuggers like [gdb](http://en.wikipedia.org/wiki/Gdb) are also used, and these often provide less of a visual environment, usually using a [command line](http://en.wikipedia.org/wiki/Command_line).