Materi Kuliah
MATRIKULASI

KOMPUTASI

dan

PEMROGRAMAN

Rhiza S. Sadjad
rhiza@unhas.ac.id
http://www.unhas.ac.id/rhiza/

TOPIK 3: PEMROGRAMAN

Topik 3

PEMROGRAMAN

TOPIK 3

PEMROGRAMAN

Programming

Dari: http://en.wikipedia.org/

Computer programming (often shortened to ***programming*** or ***coding***) is the process of *writing*, *testing*, *debugging/troubleshooting*, and *maintaining* the source code of computer programs. This source code is written in a 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* and *formal logic*.

TOPIK 3

Contents

TOPIK 3

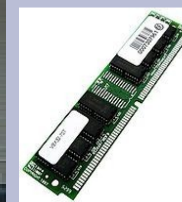
Overview

Within *software engineering*, ***programming*** (the implementation) is regarded as one phase in a software development process. There is an ongoing debate on the extent to which the writing of programs is **an art**, **a craft** or **an engineering discipline**.[1] **Good programming** is generally 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 generally **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*." However, representing oneself as a "*Professional Software Engineer*" **without a license** from an accredited institution is **illegal** in many parts of the world.[citation needed]

Another ongoing debate is the extent to which the programming language used in writing computer programs affects the form that the final program takes. This debate is analogous to that surrounding the Sapir-Whorf hypothesis [2] in linguistics, that postulates that a particular language's nature influences the habitual thought of its speakers. Different **language patterns** yield different **patterns of thought**. This idea challenges the possibility of representing the world perfectly with language, because it acknowledges that the mechanisms of any language condition the thoughts of its speaker community.

Said another way, programming is the craft of transforming **requirements** into something that a computer can **execute**.

TOPIK 3



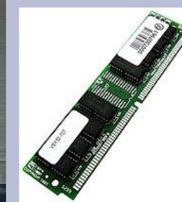
Computer Programming

http://en.wikipedia.org/wiki/List_of_programming_languages

History of Programming

The ***concept*** of devices that operate following a pre-defined set of instructions traces back to Greek Mythology, notably Hephaestus and his ***mechanical servants*** [3]. The Antikythera mechanism was a calculator utilizing gears of various sizes and configuration to determine its operation. The earliest known programmable machines (machines whose behavior can be controlled and predicted with a set of instructions) were a Muslim Scientist Al-Jazari's programmable Automata in 1206.[4]. One of Al-Jazari's robots was originally a boat with four automatic musicians that floated on a lake to entertain guests at royal drinking parties. Programming pegs and cams into a wooden drum at specific locations. These would then bump into little levers that operate a percussion instrument. The output of this device was a small drummer playing various rhythms and drum patterns. [5][6] Another sophisticated programmable machine by Al-Jazari was the castle clock, notable for its concept of variables which the operator could manipulate as necessary (i.e. the length of day and night). The Jacquard Loom, which Joseph Marie Jacquard developed in 1801, uses a series of pasteboard cards with holes punched in them. The hole pattern represented the pattern that the loom had to follow in weaving cloth. The loom could produce entirely different weaves using different sets of cards.

TOPIK 3



Computer Programming

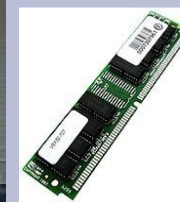
http://en.wikipedia.org/wiki/History_of_programming_languages

Contents

- ***1 Before 1940***
- ***2 The 1940s***
- ***3 The 1950s and 1960s***
- ***4 1967-1978: establishing fundamental paradigms***
- ***5 The 1980s: consolidation, modules, performance***
- ***6 The 1990s: the Internet age***
- ***7 Current trends:***

- ***Programming language evolution continues, in both industry and research. Some of the current trends include: Mechanisms for adding security and reliability verification to the language: extended static checking, information flow control, static thread safety.***
- ***Alternative mechanisms for modularity: mixins, delegates, aspects.***
- ***Component-oriented software development***
- ***Metaprogramming, reflection or access to the abstract syntax tree***
- ***Increased emphasis on distribution and mobility.***
- ***Integration with databases, including XML and relational databases.***
- ***Support for Unicode so that source code (program text) is not restricted to those characters contained in the ASCII character set; allowing, for example, use of non-Latin-based scripts or extended punctuation.***
- ***XML for graphical interface (XUL, XAML).***

TOPIK 3

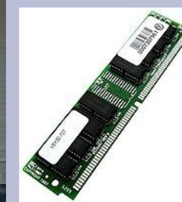


Computer Programming

http://en.wikipedia.org/wiki/History_of_programming_languages

Prominent people in the history of programming languages
John Backus, inventor of Fortran
John McCarthy, inventor of LISP
Alan Cooper, developer of Visual Basic
Edsger W. Dijkstra, developed the framework for proper programming
James Gosling, developer of Oak, the precursor of Java
Anders Hejlsberg, developer of Turbo Pascal and C#
Grace Hopper, developer of Flow-Matic, influencing COBOL
Kenneth E. Iverson, developer of APL
Bill Joy, inventor of vi, early author of BSD Unix,
and originator of SunOS, which became Solaris
Alan Kay, pioneering work on object-oriented
programming, and originator of Smalltalk.
Brian Kernighan, co-author of the first book on
the C programming language with
Dennis Ritchie, coauthor of the AWK and AMPL
John von Neumann, originator of the operating system concept.
Dennis Ritchie, inventor of C.
Bjarne Stroustrup, developer of C++.
Ken Thompson, inventor of Unix.
Niklaus Wirth inventor of Pascal and Modula.

TOPIK 3



Computer programming

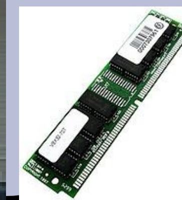
Modern programming

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

- <u>Efficiency/performance:</u> 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 and lack of memory leaks.
- <u>Reliability</u>: 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 and race conditions) and logic errors (such as division by zero).
- <u>Robustness</u>: how well a program anticipates problems not due to programmer 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.
- <u>Usability</u>: the 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.
- <u>Portability</u>: the range of computer hardware and operating system platforms on which the source code of a program can be compiled or interpreted 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.

TOPIK 3

Computer Programming



Modern programming (continued......)

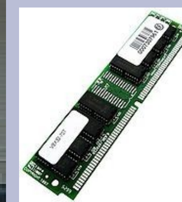
Algorithmic complexity
The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problem. For this purpose, algorithms are classified into orders using so-called Big O notation, O(n), 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
The first step in most formal software development projects is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). There exist a lot of differing approaches for each of those tasks. One approach popular for requirements analysis is Use Case analysis. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). The Unified Modeling Language (UML) is a notation used for both OOAD and MDA. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling).Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.

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 is still strong in the corporate data center, often on large mainframes, FORTRAN in engineering applications, scripting languages in web development, and C in embedded applications), while some languages are regularly used to write many different kinds of applications. Methods of measuring language popularity include: counting the number of job advertisements that mention the language[10], the number of books teaching the language that are sold (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).

TOPIK 3

Computer Programming



Modern programming (continued......)

Debugging
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 somekinds of faults because their specification does not require compilers to perform as much checking as other languages. Use of a static analysis tool can help detect some possible problems. Debugging is often done with IDEs like Visual Studio, NetBeans, and Eclipse. Standalone debuggers like gdb are also used, and these often provide less of a visual environment, usually using a command line.

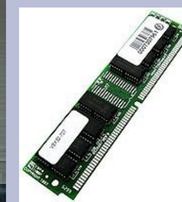
Programming languages
Different programming languages support different styles of programming (called programming paradigms). The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Ideally, 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. **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**: Get 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 libraries. Provided the functions in a library follow the appropriate runtime conventions (eg, method of passing arguments), then these functions may be written in any other language.

TOPIK 3



Computer Programming

Modern programming (continued......)

Programmers

Computer ***programmers*** are those who ***write computer software***. Their jobs usually involve:

- Coding
- Compilation
- Documentation
- Integration
- Maintenance
- Requirements analysis
- Software architecture
- Software testing
- Specification
- Debugging

References

1. Paul **Graham** (2003). Hackers and Painters http://www.paulgraham.com/hp.html. Retrieved on 2006-08-22.
2. Kenneth E. **Iverson**, the originator of the APL programming language, believed that the Sapir–Whorf hypothesis applied to computer languages (without actually mentioning the hypothesis by name). His Turing award lecture, "*Notation as a tool of thought*", was devoted to this theme, arguing that more powerful notations aided thinking about computer algorithms. Iverson, K.E.,"*Notation as a tool of thought*", Communications of the ACM, 23:444-465 (August 1980).
3. New World Encyclopedia Online Edition New World Encyclopedia
4. Al-Jazari - the Mechanical Genius, MuslimHeritage.com
5. "*A 13th Century Programmable Robot*", University of Sheffield
6. **Fowler**, Charles B. (October 1967), "The Museum of Music: A History of Mechanical Instruments", Music Educators Journal 54 (2): 45–49, doi:10.2307/3391092
7. Columbia University Computing History - Herman Hollerith
8. [1]
9. [2]
10. Survey of Job advertisements mentioning a given language

Further reading

- **Weinberg**, Gerald M., "*The Psychology of Computer Programming*", New York: Van Nostrand Reinhold, 1971

Next ....

Evaluation !!!

EVALUASI