SCITECH
RESEARCH ORGANISATION|
Volume 8, Issue 1
December 11, 2018
Journal of Information Sciences and Computing Technologies
www.scitecresearch.com/journals

Features of Programming Languages and Algorithm for Calculating the Effectiveness

Shafagat Mahmudova

Institute of Information Technologies of ANAS, Baku, Azerbaijan

Abstract

The article provides information on the basics of software engineering, programming and programming languages. Software engineering is also defined as a systematic approach to the analysis, scheduling, design, evaluation, implementation, testing, service and software upgrading. Thinking and the peculiarities of the algorithmic peculiarities are clarified, and the mechanism of their use in programming is explained. Programming theory incorporates the formal methods based on software specifications and the method based on the mathematical subjects and provides program development using mathematical symbols and ensures the accuracy to obtain the required results on the computer. The principles of using graphs in programming and dynamic programming are analyzed. The concepts of programming technology and programming languages are described. The criteria for evaluating the programming languages are identified and an algorithm is developed for calculating the effectiveness.

Keywords: Programming languages; concepts; principles; analyze; criteria; effectiveness.

1. Introduction

The main objective of the Information Society (IP) is to meet the information needs of all people, to create a social communication environment among them, and to convey their knowledge and ideas to the public. That is, information society is such a community that most of the population is engaged in the production, storage, processing and transmission of knowledge, which is the most important form of information, using modern information technologies.

Computers affect all the processes in society, including scientific research and economy. Generally, they change the way people operate, and penetrate new areas of practice. Learning new technologies and applying them to different areas leads to the development and growth of modern systems and programming languages. Software engineering deals with this area. To succeed in addressing a variety of issues, one needs to benefit from scientific innovations.

One of the main goals of modern software engineering is to develop software products (SPs) for different problem-oriented areas and to ensure their effectiveness throughout the life cycle.

Software engineering is regarded as a combination of the principles of the engineering approaches designed for the production of artifacts (produced by human beings different from the natural ones), mathematics, and informatics and computer science. Software engineering is also defined as a systematic approach to the analysis, scheduling, design, evaluation, implementation, testing, service and software upgrading.

Software engineering incorporates the following fields of knowledge [4]:

- Fundamentals of computing;
- Fundamentals of mathematics and engineering;
- Professional experience (teamwork, communication skills, ethics);
- Fundamentals of modeling (analysis, work with requirements, specifications);
- Software design (concepts and project strategies, human-machine interface design, project support);

- Software verification and attestation (testing, user interface evaluation, problem analysis, etc.);
- Fundamentals of software evolution;
- Software development processes;
- Software quality;
- Software project management (management concepts, scheduling and monitoring of the project execution, personnel and configuration management).

A theory of any subject explores the fundamental components (elements) of the subject matter by using different expressions. Based on the theory, technology uses the studied components as a result of its operation and ensures the efficiency and robustness of the processes. Theory and technology are independently developing within their own laws. A good theory enables software design and ensures its successful implementation [8].

The main theories in the field of programming are as follows:

- dual theory;
- theory of numbers;
- theory of symbols and so on.

The theory is presented and its theorems and anti-theorems for expressions are explained. Theorems and anti-theorems are verified based on the following five rules:

- **Axiom Rule**. If a dual expression is an axiom, then it is a theorem. If a dual expression is an anti-axiom, then it is an anti-theorem;
- **Evaluation procedure**. If all dual sub-expressions of all dual expressions are classified, then they are classified in accordance with the accuracy tables;
- **Completion Rule**. If a dual expression contains a dual sub-expression and all their classification methods place it in one class, then it is also in the same class;
- **Consistency Rule**. If a classified double-expression contains double-sub-expressions and only one of its classification methods is agreed, then they are also classified in the same way;
- **Instance Rule**. If a dual expression is classified, then all its copies are referred to the same classification.

The software covers the topics selected from formal computer languages, computational theories, theory of types, programming languages and artificial intelligence.

2. Fundamentals of programming theory

Programming theory incorporates the formal methods based on software specifications and the methods based on the mathematical subjects (logic, algebra, combinatory analysis, etc.), and provides program development using mathematical symbols and ensures the accuracy to obtain the required results on the computer. In this regard, many trends of the programming theory have developed, thus, a method of building the programming algorithms with the application of mathematical methods has been introduced.

The basic concept of the programming theory in one of the studies refers to the followings [17]:

- Programs;
- Implementation;
- Equivalence.

A program is a set of guidelines or commands to be performed by the computer step by step. A specific algorithm is used when designing the program. That is, the program expresses each algorithm in the way understandable for the computer. In other words, the program consists of a finite number of specifically compiled sequential commands, which transform the input data into the output data in a certain sequence. The formulas used to write the program on the computer are called programming languages.

Programming languages differ from the ordinary languages for the number of words (understandable to the translator only) and the strict rules of writing the commands.

The key elements of any programming language are: alphabet, syntax and semantics of the language.

Implementation. One of the integral parts of the program is its implementation. The implementation of the program includes the transformation of the information into a machine code and achieving the result intended in the algorithm. Numerous errors may occur during the execution. The correct result can be achieved only after the errors are eliminated.

Equivalence. If two different programs have a large number of common variables and if the performed functions coincide, they are almost equally functional.

The program comprises the topics selected from the formal computer languages and covers computational theory, theory of types, programming language design and artificial intelligence.

When designing any software, it is important to comprehend the problem first, to develop a solution algorithm and visualize it through a flowchart.

Language = {Syntax, semantic, data structure} is ultimate for programming language.

Computing and algorithms constitute the key elements of the program.

The key features of the programming language theory. Various programming languages are used for the software design of the presented issues. Programming is a process of software development for a computer.

As noted, the main condition of programming is the development of algorithms. An algorithm is a sequence of operations (stages) that are essential for the problem solution. Generally, the algorithm is a formal writing that identifies the operations required to resolve a given issue and shows a sequence in which they are performed. The process of developing the algorithms is called algorithmization [5].

Programming refers to the process of developing software in a broad spectrum. This includes the analysis and statement of the problem, scheduling and design of the program, the development of the algorithms and data structures, writing, implementation, testing (trials), documentation, configuration (arrangement), completion and accomplishment of the issue.

Before the review of the development trends in the programming languages, it is necessary to analyze the driving forces of the programming languages. These are:

- Improvement;
- Focus on efficiency;
- Consideration of the problem complexity;
- Extending the life cycle of the program, and so on.

Programming Language Theory (PLT) is a field of computer science that deals with the classification of design, implementation, analysis, programming languages and their individual features. As a field of computer science, it depends on and impacts mathematics, software and linguistics. This is an active research area of informatics, thus, the research results are published in numerous journals dedicated to PLT, including in general and technical publications [12].

PLT includes many research areas, most of which are related to programming language. Moreover, PLT uses many areas of mathematics, including computational mathematics, set theory and so forth.

The good theory of programming enables to write accurate specifications and to develop programs with the predefined specifications is provided during the implementation. As mentioned, properly compiled algorithms are essential for the development of the programs. Algorithmic thinking of producers also plays an important role in the development of algorithms.

3. Algorithmic thinking

Before interpreting the Algorithmic Thinking, thinking itself should be overviewed first [16].

Thinking has numerous definitions. Some of them are shown below.

Thinking is a complicated cognitive process that is a generalized and instrumental representation of the authorized links and relationships between the objects and events.

Thinking, unlike other cognitive processes, reflects the internal relationship and the essence of the objects and events. Without it, it would be impossible to understand the external world and benefit from its laws.

“Thinking” is the accumulation and generation of the information collected through senses and emotions in the brain. Consciousness is its core, while senses and comprehension are its perception.

The manner of thinking differs for its:

- formality;
- logic;
- clarity;
- ability to deliver any abstract idea to successive instructions;
- gradual implementation;
- and so forth.

The quality of thinking may include:

- the width of mental thought;
- the specificity of mental thought;
- the depth of mental thought;
- the agility of mental thought;
- the pace of mental thought;
- the consistency of mental thought;
- the independence of mental thought;
- and so forth.

Algorithmic thinking is defined in various ways. Some of them are given below [1].

1st Definition. Algorithmic thinking - a mental process is referred to the interpretation process. This means that even the equally accepted concepts are understood differently by different people, that is, the interpretation in the thinking process depends on a number of factors as age, education, outlook, life experience, and so forth.

2nd Definition. Algorithmic thinking - a set of ideas and perceptions aimed at the solution of the given issues, as a result of which an algorithm - a special product of human activity is generated.

3rd Definition. Algorithmic thinking is understood as a system of perceptions aimed at solving issues.

Algorithmic thinking helps to solve problems well in any sphere of the human activity [2].

Thus, algorithmic thinking is a stage-by-stage and perceived process of knowledge characterized by a consistent transition to the new knowledge.

One issue here is particularly noteworthy. If the manufacturers work together, they must understand one another's algorithm and only then develop own algorithm.

Algorithmic thinking is of great importance for mastering new knowledge and skills. Some skills are required in many spheres and may include:

- the ability of splitting up the issues into sub-issues;
- the ability of the programmer to plan the stages of the problem and his/her time;
- the ability to evaluate the effectiveness of the activity;
- the ability to search for information;
- the ability of data processing and retrieval;
- the ability to perform consistency, parallel operations, etc.

The algorithmic thinking should be trained a lot to develop.

Developing the algorithmic thinking that shapes the appropriate manner of thinking is very important for solving issues [13].

It should be noted that, in fact, the algorithmic thinking is important for each human being. Unlike classical, logical, or creative thinking, the algorithmic thinking is valid for a clearly described system.

4. Basic concepts of programming technology

The subject matter of programming technology is the process of software development.

A process is a sequence of actions to produce any result.

A model is an image or an analogue of any object [9].

Variety of completely different tools is used to build the models in different areas of human activity. Physical processes are characterized by the mathematical models, such as the differential equations systems.

Moreover, the process of program development are attempted to describe in a variety of ways. The program is written as an algorithm using programming languages.

Note that the software development on the modern technology is regarded as a parallel process interacting one another, rather than a sequential processes. This, in turn, causes different algorithmic and terminological challenges.

Various technologies are used in the course of software development. The notion "technology" in programming technology is understood as a set of production processes and a description of a set of processes requiring the development of a software system. In other words, programming technology is largely regarded as a software development tool, i.e., all the processes starting from the idea of this tool to the development of the required software documents. Each process of this framework is based on the use of any methods and means [18].

Historically, the development of programming technology can be divided into several stages:

1. First stage: a "natural" programming cycle - the absence of certain technology, the first stage is related to the emergence of ECM. It continues until the mid-60s of the 20th century. The growth of programming continued with the replacement of machine language through the assembler, and later, the subsequent reuse of algorithmic languages (Fortran, Algol) and subprograms increased the performance of the programmer. In the early 1960 s, the programming crisis broke out;

2. Second stage - a structural approach to programming appeared. This approach emerged in the 1960 s. According to the structural approach, the programs were divided into smaller sub-programs. In contrast to the previous procedural approach of partition, the structural approach required the presentation of the problem in the form of hierarchy of the sub-problems of the simplest structure. Procedural programming languages (PL / 1, Algol-68, Pascal, C) were established with the support of the structural programming principles;

3. Third stage – object-oriented approach to programming began. This stage began in the mid 80's. Object-oriented programming is defined as a complex software development technology as a set of objects. Here, the concept of class plays a major role, since the classes create a hierarchy with the inherent characteristics. Compared to the modular programming, the main advantage of the object-oriented programming is dividing the software in more natural way, thus, it significantly eases the development process. In addition, object-oriented approach offers new methods for the software development based on the mechanisms of inherence, namely polymorphism and compositions. This significantly reduces the rate of the repeated use of the codes and generates the libraries of the classes for different applications. The development of the object approach in programming technology has led to the creation of visual programming environments. Later, visual object-oriented programming languages, such as Delphi, C++ Builder and Visual C++, C # appeared. However, OOP technology has some shortcomings. They may include the dependence of the software modules on the addresses of the exported fields and methods, structures and data formats. This dependence is obvious due to the fact that the modules should interact with each other by accessing the resources.

4. Fourth stage is related to the component approach and CASE-technologies (from the mid-90s to the present time). This approach uses separate components to build software. The existing parts of the software are physically separate, thus, they interact with each other through the standardized dual interfaces. The basics for the Component approach were developed by Microsoft Company, which were used in earlier editions of Windows to create the content documents starting with Object Linking and Embedding (OLE) technology. Emergence of the Component Object Model (COM) has led to its development, and later, its distributed edition - DCOM has been widely used in software development.

Component approach was followed by Common Object Request Bracer Architecture (CORBA). This technology was developed by an analogical group of companies COM, OMC (Object Management Group). CORBA's software core is implemented for all major hardware and software platforms and provides software in a heterogenic computing environment.

The most important feature of the modern stage of programming technology is a wide application of computer technology of software systems at all stages of the lifecycle of the software.

These technologies are Computer Aided Software / System engineering (CASE) for the software development. Today, CASE technologies are supported both in the structural object and component approaches of programming.

5. Basic concepts of programming languages

Programming languages include specific elements. These key elements include the followings [17]:

- Software environment;
- Basic syntax;
- Data types;
- Variables;
- Keywords;
- Basic operators;
- Numbers;
- Characters;
- Arrays;
- Strings;
- Functions;
- Input/output.

The 5 basic elements of any programming language are:

1. Variables;
2. Control structures;
3. Data structures;
4. Syntax;
5. Tools.

Note that 2 type variables are used for the program designing.

Static variables are long-term variables available in a function or file. They are different from global variables. Static variables are very advantageous when they are needed to be written to universal functions and function libraries, since the programmers may use them for a long time.

The variable in programming becomes global when it is applicable to all software applications. The global variables can be used as an alternative to arguments transferring for the interaction between the procedures and functions.

The variable in programming is an embedded memory space.

In other programming paradigms, such as functionality and logic, the concept of variable is slightly different. The variable in such languages is designated as a name, as they may be related to a value, or may even be used as a location to store the value [11].

6. Dynamic programming

Dynamic programming is used to solve many complex tasks, since they are divided into sub-tasks, and therefore, the optimal solutions to sub-tasks are used for the optimal solution of the key task. Dynamic programming was introduced by the US mathematician Richard Bellman. The proposed dynamic programming theory has been crucial for the analysis and optimization of large-scale dynamic systems [6].

Manufacturers discovered many shortcomings, that is, errors in the programming algorithms, the elimination of which was very difficult for them. Therefore, they tried to use the capabilities of the dynamic programming, i.e., the dynamic programming managed to remove these errors. It simply corrects the errors, and consequently, the manufacturers have achieved the optimal method.

Dynamic programming includes two main elements:

Optimal sub-structure. If an optimal sub-structure is defined for a particular task, then the dynamic programming can be applied to the task too. If the task can be split up into auxiliary tasks, then the optimal solutions to the sub-tasks can be used to optimize the main task;

Memorization. If the auxiliary tasks are frequently repeated, then it is better to find the optimal solution of the task repeated only once and accept the result.

These two elements are considered fundamental in the dynamic programming.

Graphs in programming. The graphs are very difficult to encode, however, they comprise very interesting real-time software. If the graphs are designated by mathematical terms, they become clear.

The graph G (V, E) consists of two sets:

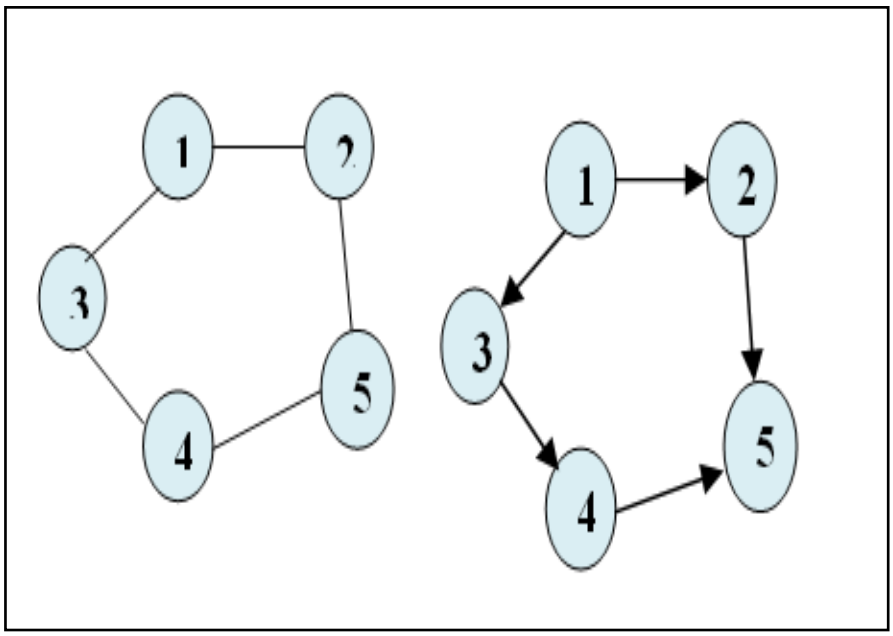
- V- set of vertices (dots);
- E- set of edges.

| V | the number of hills in the column.

| E | indicates the number of columns in the

Figure 1 (a) is a non-directional graph, shown in Figure 1 b).

V = 6 E = 5



a) b)

Fig. 1. a) undirected graph, b) directed graph

7. Optimization problem in programming

Program optimization or software optimization is a process of software upgrade to make some software aspects perform more effectively or use less resource [14]. Generally, the software may be optimized to run faster or work with other resources with smaller capacity, or to be able to consume less energy.

Optimization levels. Optimization may be implemented on several levels. As a rule, higher levels have a great impact, however, it becomes very difficult to make some changes to them later in the project; significant changes require rewriting. Thus, the optimization can be typically implemented through accuracy from lower levels to higher levels. However, in some cases, the overall performance of the program may depend on the performance of the lower levels, or small changes in early discussion of the low-level details may have negative effects. As a rule, all projects focus on the effectiveness more. Long-term projects typically have optimization periods, as the improvement of one area reveal the limitations in other areas, leading to additional costs.

Performance is a part of the software specificity, accordingly, it should be taken into account that the system can provide sufficient performance and the expected result of the performance may be achieved by taking into account optimization.

The rate of the change in the performance between the prototype and production system is subjected to the optimization accordingly, and may be significant for uncertainty and risk.

Effective development of the algorithm and the correct structuring of data play a key role in the software development. The algorithm of the task and the structure of the data are the foremost aspects affecting the effectiveness. Later modification of the data structure is more challenging that the modification of the algorithm.

Computing tasks can be solved in a variety of ways and with different efficiency. An effective case is called a reduction of the power through the corresponding functionality. For example, the review of the fragment of the program code in the programming language Pascal shows that the Goal is to obtain a sum of the set of integers from 1 to N:

```
i, S integer;
S: = 0;
N: = 10;
for i: = 1 to N;
  S: = S: + i;
Write (s);
```

This code is denoted as a mathematical formula as follows:

```
Integer S: = N * (1 + N) / 2;
Write (s);
```

An algorithmic effectiveness in informatics is the feature of the algorithm, which refers to the number of computing resources used in the algorithm. The algorithm is analyzed for the definition of the usage of resources. The algorithmic effectiveness can be considered as an analogue of engineering performance for repeated or uninterrupted process [3].

It is necessary to minimize the resources used to achieve the maximum efficiency. Nevertheless, different sources (e.g., time, environment) cannot be directly compared, and therefore, one of the two algorithms, the effectiveness of which is more significant, is effective. For example, high speed, minimum memory usage, or some other performance indicators, and so forth.

The following features should be considered in programming:

- Analysis of algorithms – the identification of the resources required for the algorithm;
- Arithmetic coding - a form of entropic coding used for the effective compression of the data of variable length;
- Associative array - a data structure that can be more effective when using Patrice trees or Judy arrays;
- Benchmark - a measurement method for a comparative execution time in certain events;
- Best, Worst and Average Case - best, worst, and average Ideas for the evaluation of the implementation in three scenarios;
- Binary search algorithm - a simple and effective method of searching for sorted arrays;
- Branch table - a way to reduce the length of the instruction path, a size of the machine code (including, the memory);
- Comparison of programming paradigms - specific characteristics of the paradigm performance;
- Compiler optimization - compiler-based optimization;
- Computational complexity of mathematical operations. The theory of computational complexity:
- Computer performance;
- Data compression - Reduced rate of data transfer and disk capability;
- Database index - a data structure that improves the pace of the data search operations in the database table;
- Entropy Encoding - used as an effective criterion to replace the lines;
- Garbage collection - automatic cleaning of the memory after the usage;
- Green computing - transition to the application of the technologies consuming less resource;
- Huffman algorithm – an algorithm for effective data encoding;
- Improving Managed Code Performance - Microsoft MSDN Library;
- Locality of reference –preventing the caching delays caused (called) by the non-local access;
- Loop optimization;
- Memory management;

- Optimization (computer science);
- Performance analysis - assessment methods of actual performance of the algorithm during the implementation;
- Real-time analysis - additional examples of the critical programs;
- Run-time analysis - an evaluation of the expected performance, and an algorithm scaling;
- Simultaneous multithreading;
- Speculative execution or Eager execution;
- Super-threading;
- Threaded code - a table of similar virtual methods or a table of branches;
- Virtual method table - Schedule of the virtual methods tables for dynamically assigned indicators for dispatching.

8. Criteria for the assessment of the programming languages and algorithm for the calculation of the effectiveness

The power, level and conceptual integrity of programming language are the main characteristics that allow comparing the programming languages and selecting the best one for the given task [10].

The power of language is characterized by the number and variety of tasks, and the algorithms. The machine language is considered to be the most powerful. Any task programmed in any programming language can be programmed in the machine language too.

The level of language is characterized by the difficulty of solving the task through this language. The more simply the problem solution is written, the more complex operations and concepts are realized, thus, the volume of programs decreases depending on the higher-level language.

The conceptual integrity of the language, in turn, is characterized by a set of concepts describing this language and includes three interconnected aspects: economy, orthogonality and uniformity of the concepts. Typically, the less the power of language, the higher its level is [7].

Each programming language has its own advantages and disadvantages. The key features of programming languages are listed below [15]:

- **Completeness** - must be followed in the programming language, i.e., it must be equipped it with the constructions to describe the syntax and semantics of that language. The completeness of the language provides a description of a particular subject matter, such as the working process of the programs, with the help of language tools. The completeness is a system feature. The marginally complete system is not split up into parts, although, in a well-developed system, the elements complement and help each other replacing the functions and the elements of the external systems;
- **Clarity** - the structure of the programming language must be clear to the programmer;
- **Reliability** – the rate of the automatic error detection when the software is implemented or translated. Reliable programming language detects the errors during the translation of the program, rather than its implementation. The sooner the errors occur during the implementation, the much cheaper the project will be;
- **Elasticity** - gives the programmer all the tools to apply external programs to express the process when writing the program. Flexibility of the language provides in the application area only in the application area;
- **Simplicity** - allows to understand and memorize its semantic structure;
- **Natural** - an important characteristic of the programming language, since the concepts, standards, and custom signs in the terminology of the manufacturer have an intuitive similarity to those in languages;
- **Mobility** - capabilities of the programming language allow the program to be transferred from one platform to another regardless of the device resulting in reduced costs;
- **Valuable** - includes language learning, software development and translation, implementation, and accompaniment costs.

The effectiveness can be defined by using the basic criteria for evaluating programming languages. It has different ways. Each criterion has its own individual indicators. These criteria are denoted as follows:

T - Completeness,

A - Clarity,

E - Reliability,

EL - Elasticity,

S - Simplicity,

TE - Natural,

M - Mobility,

D - Valuable.

One of these criteria is selected and shown as an individual set $C_1, C_2, C_3, \dots, C_n$ here, C_1 is a key indicator, and conditions are assigned to others.

$$C_i \le C_{ie} \ \ (i = 2,3, \dots, n)$$

Here C_{ie} is the value of the i- th indicator.

If the fullness (T) is principle, the remaining criteria will be limited, and then the effectiveness of the programming language will be as follows:

$T \to max, A \le A_e, E \le E_e, EL \le EL_e, S \le S_e, TE \le TE_e, M \le M_e, D \le D_e,$

Individual indicators of the criteria are listed according to their importance level. As noted, C_1 is the most important, and C_n is the least important indicator.

Minimum value of C_1 is found - min C_1. C_1 is compromised and $\Delta C_1$and min $C_1 + \Delta C_1$

$C_1 \le \min C_1 + \Delta C_1$is obtained.

The same rule is applied to C_2 and consequently

$C_2 \le \min C_2 + \Delta C_2$

.....

$C_{n-1} \le \min C_{n-1} + \Delta C_{n-1}$ is obtained.

In this case, the options for the programming language and the prefix values are considered to be final.

This rule applies to other indicators and ultimately:

$K_1 = D/T \to min$

$K_2 = A/T \to min$

$K_3 = E/T \to min$

$K_4 = EL/T \to min$

$K_5 = S/T \to min$

$K_6 = TE/T \to min$

$K_7 = M/T \to min$

K_i shows the criteria. Criteria can be calculated in this way, considering each of these indicators as essential.

$$K_{emax} = (C_1, C_2, C_3, \dots, C_n) / (L_1, L_2, L_3, \dots, L_n) \to max \quad (1)$$

$C_1, C_2, C_3, \dots, C_n$ – are individual indicators, so you need to increase and $(L_1, L_2, L_3, \dots, L_n)$ is lower the price. In the formula (1), the criterion of effectiveness is max.

$$K_{emin} = (L_1, L_2, L_3, \dots, L_n)/(C_1, C_2, C_3, \dots, C_n) \to min \quad (2)$$

In the formula (2) the minimum is approached.

The criteria for effectiveness

$$K = \sum_{i=1}^{n} \propto_i C_i \quad (3)$$

$$0 \le \propto_i \le 1, \ \ \sum_{i=1}^{n} \propto_i = 1 \quad (4)$$

it can be shown that,

$\propto_1, \propto_{2,\ldots}, \propto_n,$ show the weight coefficients. They can be negative or positive. The positive coefficient is designated for those indicators which should maximize and minimize the negative. The same can be applied to other indicators.

9. Conclusion

Modern stages of programming need to be thoroughly studied, and new methods of the software development should be developed based on the knowledge gained. In this regard, manufacturers should use a new methodological system using the algorithmic thinking. It should be noted that the new methods to be used will increase the quality of software.

References

[1] Stas, A. N. and Dolganova, N. F. (2012). «Развитие алгоритмического мышления в процессе обучения будущих учителей информатики». *Журнал вестник Томского государственного педагогического университета*, стр. 241-244.

[2] Bovi, P. (2010). «Алгоритмическое мышление». *Информатика*, стр. 1-5.

[3] Kriegel, H.P., Schubert, E. and Zimek, A. (2016). "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?". *Knowledge and Information Systems*, Vol. 52, No. 2, pp. 341–378. DOI: 10.1007/s10115-016-1004-2.

[4] Boyko, N. I. and Zverintsev, M.Ye. (2007). *Рекомендации по преподаванию программной инженерии и информатики в университетах и др.* Интернет-университет Информационных технологий.

[5] Benjamin, C. P. (2002). *Types and Programming Languages*. MIT Press, Cambridge, MA, USA.

[6] Tafhim, U.I. , (2017). *Elements of Dynamic Programming*. International Islamic Universiti, Chittagoing.

[7] Sebesta, R.U. (2011). *Основные концепции языков программирования*. М.: Вильямс. (2011).

[8] Neal, N. and Sheryl, S.,(2018).*Programming Language Theory and Practice*. Class Standing: Sophomore–Senior.

[9] Gromov, Y.Y., Ivanova, O.G., Belyayev, M.P. and Minin, Y.V. (2013). *Технология программирования.* ФГБОУ ВПО «ТГТУ».

[10] Opaleva, E.A. and Samoylenko, V.P. (2005). *Языки программирования и методы трансляции*. СПб: БХВ-Петербург.

[11] Trevor, G. (2012). *Programming 101 – The 5 Basic Concepts of any Programming Language*

[12] Robert, H. (2016). *Practical Foundations for Programming Languages.* Cambridge University Press.

[13] Rogozhkina, I.B. (2012). «Развивающий эффект обучения программированию». *Психология. Журнал высшей школы экономики*, Том. 9, № 2, стр. 141–148.

[14] Suejb, M., Sabri, P., Alecio, B., Joanna, K. and Ivona, B. (2018). "Using Meta-heuristics and Machine Learning for Software Optimization of Parallel Computing Systems: A Systematic Literature Review". *Computing*, pp. 1-44. https://doi.org/10.1007/s00607-018-0614-9.

[15] Tolstykh, T.O. and Dudareva, O.V. (2011). «Критерии и методы оценки эффективности деятельности предприятия». *Журнал. Вестник Воронежского государственного технического университета*, стр. 98-102.

[16] Shirochin, V.P. (2004). *Архитектоника мышления и нейроинтеллект. Программирование доверия в эволюции интеллекта.* Юниор.

[17] Gilyarevskiy, R.S., Nazarov, S.V., Belousova, S.N., Bessonova, I.A. and Gudyno L.P. (2012). *Введение в программные системы и их разработку.* (ИНТУИТ) Москва.

[18] Yekaterina, L. and Vladimir, P. (2016). *Прикладные и теоретические методы программирования*. ИНТУИТ Москва

Author' Biography

Shafagat Mahmudova has established multi-purpose data base control systems "Polyclinic", "Education", "Human resources", corporate information system "Training Innovation Center", "Recognition" automated identification system, "Project Registration" information retrieval system on the Journals of Problems of Information Technology and Information Society.

Sh. Mahmudova defended the thesis on the "Development of methods and algorithms for human face recognition on the basis of photo-portraits" in the specialty 3338.01 - "System analysis, control and information processing" and gained PhD degree in Technical sciences. She is associate professor.

Currently she conducts research on the "Development of methods and algorithms for the racial identity of human face on the basis of photo-portraits".

She is the author of 47 articles and 43 theses. 51 of them were published in the international journals.

From 1998 to 2011, she taught "Informatics" class at "Applied Mathematics" faculty of Baku State University.

Sh.J.Mahmudova was elected deputy editor-in-chief of International Journal of Intelligent Information Processing (IJIIP), and a member of editorship of Gconference.NET portal.

Sh. Mahmudova was elected deputy editor-in-chief of International Journal of Intelligent Information Processing (IJIIP), and a member of editorship of Gconference.NET portal.

Sh. Mahmudova was elected a reviewer of International Journal of Automation and Power Engineering. The journal is published by the Science and Engineering Publishing Company (Riley, Indiana, USA).

Sh. Mahmudova was elected a reviewer of "Pattern Recognition", Journal of Control Engineering and Technology (JCET) and "British Journal of Applied Science & Technology".

He was elected as a member of the International Association for Pattern Recognition (IAPR) and American Association for Science and Technology (AASCIT).

She teaches at the "Training Innovation Center" of ANAS Institute of Information Technologies.

Currently works as a chief engineer of the Institute.