Available online at http://jess.esrae.ru/

"Journal of Economics and Social Sciences"

Mechanism for nominating operators in Objective-C and Swift programming languages

Tomsk Polytechnic University

Gleb Gorkoltsev [a], Irina Kraevskaya [a]

[a] School of Engineering Entrepreneurship, Tomsk Polytechnic University

Abstract

The paper examines main features of nomination mechanism for operators in Objective-C and Swift programming languages. Current study analyzes goals and processes of programming languages creation, criteria for nomination of operation, and the principles of nomination. Special complexity of nominating operators consists of their adaptation to the host language system. The programming language is one of the least studied objects in the field of technical translation. Translation of such texts is a rather complex sphere of creative activity, and professional performance of work is guaranteed only by a qualitatively prepared specialist. As a preliminary stage of translation, the paper deals with the principles of the nomination of operators in programming languages, in order to ease the understanding of original texts. The study provides a detailed component analysis, which makes clear main objects and phenomena laid down in the principle of nomination of a specific operator. Nowadays programming languages is an integral part of modern engineering as they allow creating useful software and research platforms. Social responsibilities in modern engineering to a certain extent are defined by a good knowledge of programming languages.

Keywords: Programming language, nomination, operator, technical translation, translation studies;

1. Introduction

The rapid development of the IT industry in the 21st century led to the increase in the level of demand for highly qualified specialists in this field, and also influenced on the growing interest among experts and beginners both. Furthermore, it also increased the interest in programming techniques from simple "landing-page" websites to full-fledged mobile applications, which, in its turn, are gaining popularity every day. One of the important key points that gave the development of the IT sphere is Steve Jobs' statement on the Apple presentation about the opening of the iOS system for developers. After his speech, thousands of people showed interest to learning various programming languages.

Nowadays, there is a large database of teaching materials for programming. This database includes not only printed publications, but also webinars, video courses, and specialized websites that are developed not only by experienced IT companies, but also by world leading universities. Nevertheless, many aspects of modern programming languages cause difficulties in perception and understanding for beginners. The most significant difficulties are the absence of interpretations of syntactic constructions, the absence of explanations for the names of operators and, taking into consideration the teaching methodology, the impossibility of explaining

historically developed things, which are contrary to logic from the Russian user's point of view. The problems mentioned above had no other solution than mandatory and compulsory memorization without understanding.

In this regard, the relevance of the study on the mechanisms of nominating operators (operators are the speech units of the Objective-C and Swift programming languages) is determined by the challenge to theoretical knowledge from applied studies such as terminology, terminography, translation studies, and teaching methodology. In addition, this study is also relevant for the international exchange of experience in this field, in order to enhance the effectiveness of communication processes in the IT industry.

1.2 Materials and methods

During the primary analysis, it is noted that there are practically no terms in the lexicon of the programming languages. Instead of terms, often used common words of modern English are applicable. In this regard, to meet the criterion "often used" we analyzed the Corpus of Contemporary American English, the archive of the most used words in American English [5]. After analyzing this Corpus, it is revealed that 5000 words are included in the range of widely used words. In addition, an archive of 86800 words was analyzed, this archive is created by Jonathan Harris, and the words are arranged in the order of their frequency of use [4].

In order to study selected units, the method of scientific description, onomasiological approach, contextual analysis, analysis of the word internal form, as well as component and quantitative analyzes are used.

2. Discussion

Before the direct analysis of operators, it is necessary to consider the processes of creating a programming language. In the era of open-source an unlimited number of authoring programming languages appeared, the creators generally pursued the following goals:

- Widening of own tools (unwillingness to wait for an updated version of the used programming language);
- Specific tasks that are not covered by used programming languages. For example: llvm, parrot, etc. for front-end and back-end;
- The practice of code creating;

The latter motivation is of no value, for this research and for the IT community both, as it pursues subjective goals, such as widening one's own knowledge in the field of code creation, etc. Following such subjective goals leads to a lack of logic in the names of operators in authoring programming languages. After determining a goal, the process of creating a programming language begins. This process has following steps:

- Identification of functional in the terms of objective or subjective needs;
- Writing an operation algorithm;
- Selection of names for speech units (operators);

The process of writing one's own programming language is a more complex scheme. Due to the lack of interest of this research in technical aspects, such items as choice of interpreter (parser) and compiler, choice of programming language for the basis, platform choice, choice of paradigm, etc. were omitted.

In programming languages developed by large companies (Oracle, Apple, Microsoft, etc.), logic and motivation in the selection of names for operators is an important requirement. Based on the preliminary analysis of the major programming languages, the following criteria were formulated, which should be met by the nomination processes of operators:

- Pithiness;
- Easy to remember;

From the previously mentioned scheme for creating programming languages follows that for the analysis of the names of operators an onomasiological approach is required. Onomasiological approach involves studying by the model "reality (denotat, referent)" – "meaning (signification)". It means a direction from the object or phenomenon to the thought of this object or phenomenon and to their designation by language means [2].

After identifying the object or phenomenon itself, a component analysis was conducted, in order to clearly reflect the thoughts about the object or phenomenon. Moreover, the component analysis also allows comparing selected items according to established criteria. For the component analysis, the following scheme was developed:

1. Identifying the meaning of a word;
2. Establishment of the semantic component;
3. Interpretation reflecting the structure of meanings;
4. Reviewing the word in context;

It should also be clarified, that the semantic components are conditionally divided into the following classes:

- object (animate and inanimate objects stone, tree, man);
- action (actions and processes to run, to think, to turn black);
- abstraction (quality and quantity red, plump, many);
- relation (relations between objects, actions, abstractions coordination, simultaneity, attribution, part-whole, cause-result).

All operators in programming languages are unique in order to avoid repetition in the code and incorrect interpretation of program whole blocks by the compiler. We have also made a list, in which all the operators are divided into groups by their functions.

Table 1.

Name of group	Example
Preprocessors	#define; #if;
Storage handles for a class	Auto; Const; Volatile;
Type operators	_Bool; _Complex; _Imaginary;
Cycle operators	While; Do; Break; Continue;
Data types	Int; Float; Double; Char;
Auxiliary initializers	Init; Struct;
Exception operators	@try; @catch; @throw;
Making decisions operators	If; else; else-if switch;
Synthesized access methods	@interface; @property; @implementation; @synthesize; @end;

One operator from each group is clarified in order to demonstrate the general principle of analysis, which is given in Table 2.

Table 2.

Name of operator	Lexical meaning	Semantic component	Interpretation	Context
#Define	definition; to establish; to describe [1]	action	explanation (formulation), explaining, revealing [3]	Your duties are clearly defined in the contract [1]
Const	constant [1]	abstraction	invariably and	The temperature

(constant)			equally acting, unchanging [3]	remained constant. Machines that are in constant use [1]
_Imaginary	imaginary, supposed [1]	abstraction	contrived, inoperable [3]	The story takes place in an imaginary world [1]
Do	to do (the nature of the action is often not defined), to achieve, to complete [1]	action	to work, to produce, to perform, to exercise, to practice; to act [3]	She knew what she was doing; we did it [1]
Int (integer)	integer [1]	abstraction	single, indivisible number [3]	The numbers -5, 0, and 3 are integers [1]
Init	initialization, to initialize, initializer [1]	object	to prepare for use; to set initial values [3]	Initializing the system [1]
@try	to try; to endeavor; to sample [1]	action	to try to do something [3]	Tried to open the window but couldn't. I tried that recipe you gave me last night [1]
If	if; every time when [1]	relation	In the case when... (in the main sentence there may be a conjunction of "then" or "so") [3]	What will we do if this doesn't work? If you eat up all your dinner you can have some chocolate [1]
@Interface	interface, interaction area [1]	object	a system of unified communications and signals, through which the devices of the computer system are connected to each other. [3]	a simple user interface; the interface between technology and tradition [1]

Analyzing operators from the group **"preprocessors"**, it is worth noting that the sign "#" does not play a role in determining the meaning of the word. It is only an identifier that allows all operators with this symbol to be assigned to a group of preprocessors.

The "#Define" operator allows predefining a name and value (for example, TRUE = 1 or FALSE = 0), and assign constants using the same action principle. So in the future, declaring a new variable, a developer will only need to specify the name "true" or "false" for the unchanged value. This word is a verb exclusively, therefore in any context it will be presented as a verb.

Storage handles for the class provide the compiler with information about the intended use of the variable in the program. In both cases, listed in the "Context" column, the meaning of the word "constant" is denoted as the quality of stability, immutability. [4]

Type operators, like preprocessor operators, have their own identifier "_", which does not affect the meaning of the word, but only allows them to be assigned to a specific group. The "_Imaginary" operator gives the compiler information that the variable will change, and its values will be represented as irrational numbers (invalid numbers). In the above context, the word "imaginary" did not change the class belonging and its value as well. [4]

Cycle operators are regularly used in the programming process to organize multiple executions of instructions set and from a pragmatic point of view (for example, the computation speed). To specify the requirements of the person to the computer, operators were formed, in order to explain the required actions, the time interval, etc. The "Do" operator is placed after the conditional, which sets the parameters. "Do" is responsible for the commands of performing certain actions for a certain period of time.

In programming languages **data** is stored in three different **types**: int, float, and char. These operators store different types of numerical values.

The "Int (integer)" operator stores data in the form of integer values (1, 2, 3, -1, 6, etc.). Due to the specialization of this word (it is used only in mathematics and IT sphere), there are no other situations in which the meaning of the word can change; so the semantic component also remains unchanged.

Auxiliary initializers are used in both programming languages to provide classes with additional properties (setting additional initial values) due to the fact that by default classes have insufficient number of properties. As this type of operator is auxiliary, so its use in the program is not mandatory. Despite the fact that initialization is considered an action, in the already translated official textbooks on Objective-C and Swift "init" is called as "initializer". Self-correct interpretation is impossible because of the compression of the name.

To avoid errors during the compilation of a program, a developer sometimes needs to create an exception that a computer will skip instead of displaying an error on the screen and interrupting the assembly and compilation of the application. **Exception operators** have their own identifier "@" similarly to preprocessors and data types. The "@try" operator contains the first exception, relatively speaking, makes an attempt (tries), if a compiler prints the following operator, other exceptions are checked with "@catch" operators. In the second example given in the "Context" column, the verb "try" is used in direct meaning of "taste", and in indirect meaning – "get new experience". According to English features, the noun "attempt" is considered to be the object of the action "try", that is why semantic category remains unchanged. Moreover, there are no equivalents to the word "try", so the sematic component also cannot be changed into relation or abstraction.

Making decisions operators. Logic is one of the basic principals in any programming language. For the computer logic is regarded as a statement "true – 1, false - 0". On the basis of this logic the command "If, then…else" was implied. The "If" operator is the first element in the command mentioned above. This operator function is making a clause for a certain part of the code, which will work in case this clause is fulfilled. Otherwise, the compiler ignores this part of the code.

Synthesized access methods are the group of operators, which are necessary for access to data, properties, attributes and other array elements. In this area, attributes and operations are identified for each class. This section is located in the beginning of a program.

The "@Interface" operator determines basic elements that are essential for making a connection between a human and a computer.

3. Results

After the component analysis speech units were divided into groups according to their semantic components:

- Object: Char, init, int, interface;
- Action: Define, do, break, continue, try, throw, catch, switch, implementation, synthesize;
- Relation: If, while, else;
- Abstraction: Auto, const, volatile, Boolean, complex, imaginary, float, double, property;

During this study, 30 most frequent operators were selected, and after component analysis were divided into 4 groups to identify nomination trends in programming languages. It can be concluded that in the nomination there are more words belonging in the aspect of the semantic component to the classes “abstraction” (9 of 30) and “action” (10 of 30). The predominance of these word groups can be explained by the fact that programming languages mainly consist of instructions to the computer, as well as properties and various parameters that are assigned to an object in the code (from a variable to whole objects).

It is worth noting that the nomination of the operators responsible for the indications occurs solely through the formation of the verb: cause-and-effect relationships involved in logical schemes and in cycles are adverbs, types of variables are nouns, and one of the largest groups is adjectives that denote properties.

Also, with a few exceptions, the nomination of operators in programming languages is inherent in pithiness and straightforwardness. It means that general information about operator functions can be easily get from its name. Exceptions are complex and special operators. Complex operators work with other ones to perform certain functions (for example, if ... else-if ... else; interface ... implementation, etc.).

4. Conclusion

Information technology is a young and not well analyzed sphere in terms of translation studies. For successful writing of the program code, it is necessary to fully understand the meaning of the operators, therefore, so we consider that programming is inextricably connected with translation activity. In this paper we made an attempt to actualize and explain the nature of the nomination and use of a specific word and expression related to programming operations. Of course, this study focuses particular aspect of translating IT texts problems. IT sphere demands further study, and opens possibilities of comprehensive development of its vocabulary to linguists-translators. Results of the study can be applied not only in information technologies, but also in related disciplines.

References

1. Cambridge Dictionary [Available at: http://dictionary.cambridge.org/dictionary/english/] [accessed 02/02/2018]
2. Kosova, V.A. (2008). Onomasiological approach as a basis for the study of the word-formative categorization of reality. Kazan University Scientific Notes. Humanities Series. № 6. [Available at: https://cyberleninka.ru/article/n/onomasiologicheskiy-podhod-kak-osnova-issledovaniya-slovoobrazovatelnoy-kategorizatsii-deystvitelnosti] [accessed 02/02/2018].
3. Oxford Dictionary [Available at: https://en.oxforddictionaries.com/] [accessed 02/02/2018]
4. Word Count [Available at: http://wordcount.org] [accessed 02/02/2018].
5. Word Frequency Corpus of Contemporary American English [Available at: http://www.wordfrequency.info/free.asp?s=y] [accessed 02/02/2018]..