PROGRAM FOR WORKING WITH ELECTRONIC DICTIONARIES

D. Melikdzhanian

Georgian Technical University
77, M. Kostava Str., Tbilisi 0175, Georgia
davidmelikdzhanian@gmail.com

(Received 15.05.2019; accepted 11.10.2019)

Abstract

The original, simple and convenient computer program developed by the author for working with electronic dictionaries is offered. Its advantages in comparison with other similar programs are noted. Rules of work with the program and its technical features are described. The used algorithm of a fast search of words and phrases is presented.

Keywords and phrases: Object Pascal, Lazarus, program, dictionary, translation, efficiency, formatted text, encoding, algorithm, bisection method, fast search.

1 Introduction

Electronic dictionaries are now a powerful tool of a fast search for necessary information and an important alternative to "traditional" dictionaries in which information is presented in the form of a continuous text in printed books.

There is a number of different electronic dictionaries (see, for example, [1, 2]), among them there are both famous and popular dictionaries and little-known ones. Each of them has some advantages and disadvantages in comparison with others.

While assessing the quality of work of a concrete electronic dictionary, it is necessary to consider two factors independently of each other: on the one hand, how the used computer program is effective and convenient for work; on the other hand, how each text of connected to this program dictionary is successfully compiled. We often hear such argument which is an alleged evidence of an indisputable advantage of the ABBYY Lingvo program: "The English-Russian dictionary connected to this program contains more than a million words". Meanwhile, this statement tells nothing either about

the efficiency of performance of this program, or about convenience of its use.

It can be said about the program described in the present work that a possibility for the user to rather easily create and connect any concrete dictionary to this program (see below) allows to consider characteristics and features of the work of the program as the main criterion of the level and quality of the offered product.

As for the texts of the dictionaries connected to the program, – from our point of view, the idea that the more translatable words there are in the dictionary, the better it is, is a delusion. Sometimes it turns out that the dictionary with an unreasonably exaggerated text contains exotic words and phrases which practically no one uses and which only distract the unaware user who is looking for concrete information in the dictionary (not to mention useless expenditure of computer memory).

2 Description of the Program

The program "Phrase" represents a simple, convenient means for working with electronic dictionaries. The user can type an interesting for him or her word or phrase by means of the keyboard or select it from the list and immediately obtain the text with explanation of the specified concept.

The explaining text may consist of the fragments presented by different styles and different colors, may contain simple mathematical formulas, and can alternate with figures (see fig. 1).

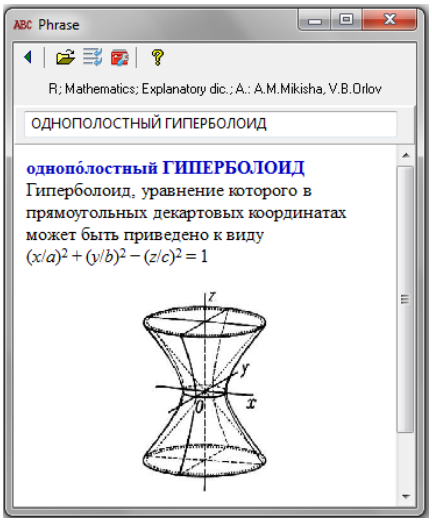

Figure 1: Mode of operating with two dictionaries (with bookmarks).

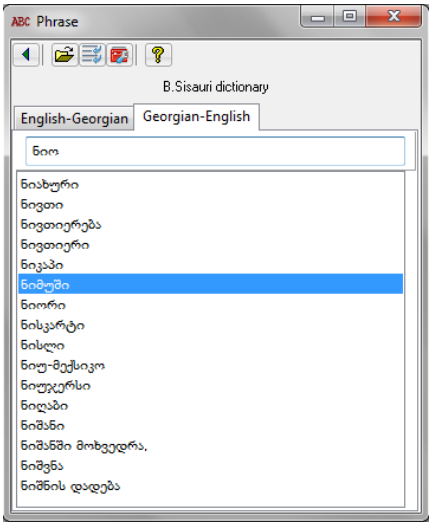

Figure 2: Mode of operating with two dictionaries (with bookmarks).

Two modes of operating with the program are possible: the mode of operating with a single dictionary and the mode of operating with two dictionaries. In the first of these modes there are no bookmarks on the main form of the application (see Fig. 1). In the second mode there are two bookmarks on the main form (see Fig. 2), by means of which it is possible to switch between two dictionaries making a logical pair (for example, between the English-Georgian and Georgian-English dictionaries).

The text of each concrete dictionary used by this program is contained in a file with extension ".phr", ".xphr" or ".yphr". In the course of working of the program the user can load any such file as well as during the work with the text or graphic editor he or she can load the edited file. There is also a possibility to load two dictionaries making a pair at once.

In comparison with other electronic dictionaries, this program has the following favorable to user distinctive features:

- This application doesn't require special installation in Windows. It, in particular, gives the user a chance to run this program from a flash drive when he or she does not work at his or her personal computer, for example being in an Internet cafe, on a business trip and also in a house or in an office of his or her friend.

- Each of the loaded dictionary files can be easily edited (revised and supplemented).

- All files necessary for working of the program take rather small space on a drive; in fact, sizes of files are minimized at a high speed of the

work of the program. It gives the user a possibility to write to a flash drive a big quantity of loaded files-dictionaries without being afraid of overflowing this drive.

The main form of the described program contains three window components for viewing and editing a text (see Fig. 3):

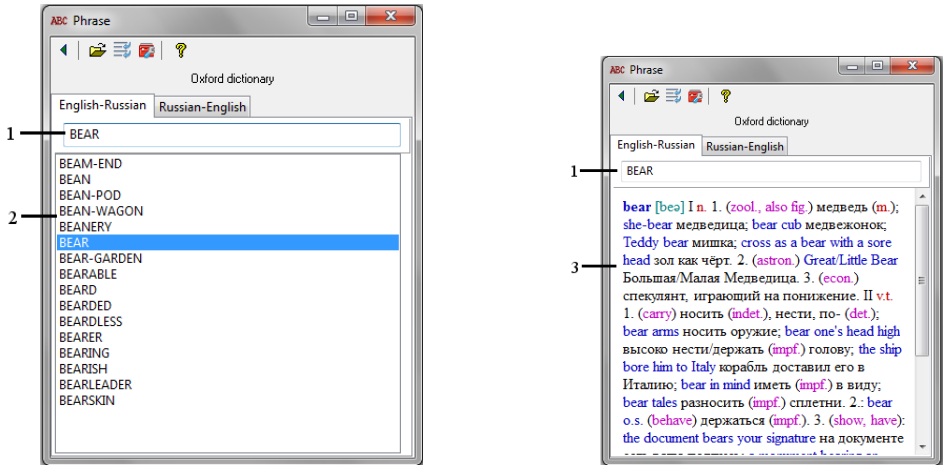

Figure 3: Window components of the program "Phrase"

1) Window for input of one line in which the user types interesting for him or her initial phrase (consisting of one or several words) from those which exist in the loaded dictionary; it is also possible to type several leading symbols of the initial phrase and then select this phrase from the list which appears in the second window;

2) Window for selection of an initial phrase from the list; this list contains 16 lines from the loaded dictionary which are grouped around the typed text in the first window at their sorting in alphabetical order;

3) Window for presentation of the multiline text explaining the meaning of the specified initial phrase.

Only one of the latter two windows is visible on the screen at every moment of working of the program, depending on a mode of working. It is made for the purpose of economy of space on the computer screen.

The second window becomes visible as soon as the first window appears in focus. The third window becomes visible when pressing the ENTER key and at double click of the mouse on one of the lines of the list in the second window.

The toolbar of this program (see Fig. 4) contains several buttons serving for selection of various commands.

The button "Exit" serves for completion of work of the program.

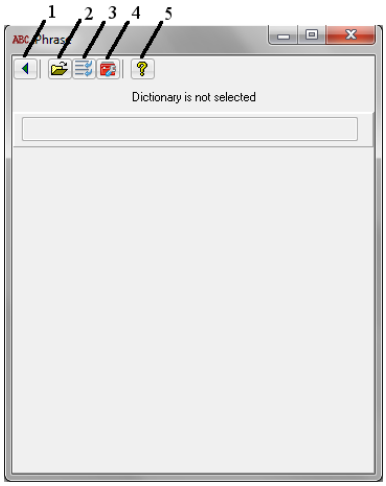

Figure 4: Buttons on the program toolbar: 1 – Exit; 2 – Open; 3 – Open from the list; 4 – Tuning; 5 – Help.

The button "Open" allows to select a concrete dictionary or a pair of dictionaries for working. For this purpose, it is necessary to load any file with extension .phr or .xphr which is in any directory on any disk driver available to the computer. When pressing the button, the standard dialogue of selection of the file used by an operating system appears on the screen.

When making a selection of the file with the extension .phr the mode of working with a single dictionary, to which the chosen file corresponds, is automatically established in the program. When selecting the file with the extension .xphr the mode of working with a pair of dictionaries is automatically established in the program; together with the file having the extension .xphr the file with the same short name having the extension .yphr is loaded (it should exist in the chosen directory). These files correspond to two dictionaries making the pair.

The button "Open from the list" (it is used more often than the button "Open") allows to select the file from the beforehand prepared list (see Fig. 5). This list is taken from the text file "phrase.ztxt" which should be in the working directory of the program. Every line of this list should contain the name of the concrete file with the extension .phr or .xphr together with the path to it, so that one would not have to look for this file on the computer.

The button "Tuning" calls the dialogue (see Fig. 6) designed for selecting a text file from the list corresponding to some dictionary, from the specified directory; these are the files with the extensions .phr, .xphr and .yphr. It is possible for the chosen text file:

- to check correctness of its structure – in this file the translated lines

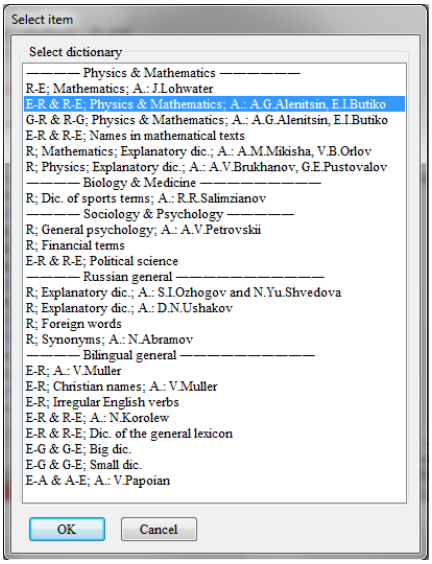

Figure 5: Selection of the dictionary from the list.

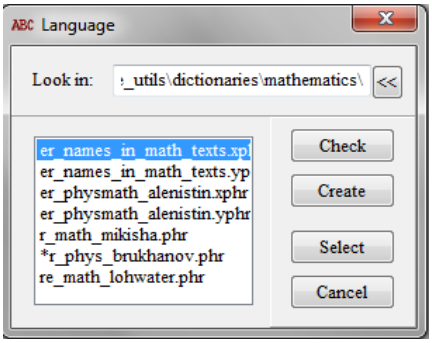

Figure 6: Checking, preparing and loading the dictionary from the specified directory.

should be located strictly in alphabetical order; the command is called by the button "Check";

- to create the auxiliary binary file; the command is called by the button "Create";
- to load the corresponding dictionary or a pair of dictionaries if these dictionaries are available for loading; the command is called by the button "Select".

In the list the names of the text files for which auxiliary binary files do

not exist are noted at the left by the symbol "*"; each such text file cannot be loaded until the auxiliary binary file is created for it.

The button "Help" displays the text from the help file. It is the HTML-file which is loaded into the browser installed by default.

3 Some Technical Features of the Program

The described program is written in the programming language Object Pascal by means of the integrated development environment Lazarus. At this stage the program can work in the operating system Windows. In prospect it is also proposed to write this program for the operating systems Linux and Android.

The whole text of each dictionary connected to the program is contained in the file with the extension .phr, .xphr or .yphr; these files do not differ from each other by their structure, and the difference of their extensions is connected only with the question of how the dictionary is used in the program – separately or together with some other dictionary.

Each of these files is a text file, some features of which are described below (see Section 3). If the text of the dictionary contains letters of any alphabet not coincident with Latin letters then the Windows encoding is used for the corresponding symbols in files. In the course of working of the program coding of such symbols automatically changes to UTF8. It is made specifically for the purpose of economy of memory of the computer; otherwise, for example, each letter of the Georgian alphabet would occupy not one but three bytes.

In the files with the dictionary texts the LaTeX commands are used for detaching various fragments of the text by different styles and different colors, for representing mathematical formulas, for formatting paragraphs and for inserting figures.

At each loading of a file with the extension .phr simultaneously with it the auxiliary binary file with the same short name and the extension .zbin is loaded. Similarly, at each loading of a file with the extension .xphr or .yphr simultaneously with it the auxiliary binary file with the same short name and with the extension .xbin or .ybin is loaded.

Each auxiliary binary file contains the sequence of integers; each such integer r_j (with number j) represents the byte number in the text file, from which j th translated line starts. If the auxiliary binary file is absent for any dictionary then it may be created (provided that the existed text file has the correct structure).

During the work of the program there is no need to use the keyboard layout pattern switch set in the operating system; such switch (Language

bar) may be absent at all in the user's computer. The program independently carries out automatic replacement of codes of the entered symbols taking into account national languages used in the selected dictionary.

Run of the program is carried out by means of the utility "phrase.exe". It is possible to specify one of the files which will be loaded (together with the auxiliary files) for working with the concrete dictionary or with a pair of dictionaries, as the parameter for this utility.

When using the described program in the operating system Windows it is possible to tune this operating system so that the utility "phrase.exe" corresponded to the files with extensions .phr, .xphr and .yphr.

4 Algorithm of search of words and phrases in a dictionary

In this section algorithms used in the described program and in earlier created packages of applied programs [3] are described.

1. Search of words in the electronic dictionary. Let there be two functions $S(j)$ and $\widetilde{S}(j)$ defined for the integer variable j from the range $(1, ..., N)$; the value of the function $S(j)$ is the text line from the dictionary with the number j, the result of the translation of which is the value of the function $\widetilde{S}(j)$. The number N represents the quantity of words and phrases in the considered dictionary. Two lists of text lines $[S(j)]_1^N$ and $[\widetilde{S}(j)]_1^N$ can be considered as components of the electronic dictionary which is being discussed in the present section.

For the convenience of work the dictionary should also be allocated by a system of search that represents the possibility for fast determination of the number of a line from the list $[S(j)]_1^N$ coinciding with the given line s (if such line exists in the list). Such system of search, obviously, also allows for the quick determination of the result of the translation of line s.

For any two text lines s_1 and s_2 it is possible to define the comparison operation $s_1 < s_2$; the result of this operation is True if when sorting in alphabetic order the line s_1 is arranged before the line s_2.

Such operation of comparison for lines is available in a ready form in the programming language Object Pascal; in the language Java the user should independently write the corresponding function.

Everywhere below it is assumed that in the list $[S(j)]_1^N$ lines are arranged in alphabetic order; this means that the function $S(j)$ is decreasing.

The offered method for finding the line s in the list $[S(j)]_1^N$ is similar to the bisection method used for the solution of the numerical equations

(see for example [4]). The essence of the offered method consists in the following.

The "big" list $[S(j)]_1^N$ can be divided into two "small" lists $[S(j)]_1^n$ and $[S(j)]_{n+1}^N$ in each of which the quantity of lines is approximately equal to $N/2$; here, $n = \mathcal{R}((1+N)/2)$; $\mathcal{R}(x)$ is the integer function the value of which is the value of the variable x approximated to the nearest integer. The determination of which of the two "small" lists the required line s belongs to does not represent a problem, as the lines $S(j)$ are arranged in alphabetical order. Thus, the quantity of lines in the list in which the given line s is searched can be reduced twice. This process can be continued until there is only one line in the list in which the line s is searched.

In the offered algorithm auxiliary integer variables $J_{\rm lo}$ and $J_{\rm hi}$, auxiliary lines $s_{\rm lo}$ and $s_{\rm hi}$ and the auxiliary logic variable A are used. At the beginning, the variable A assumes the value of False, the variables $J_{\rm lo}$ and $J_{\rm hi}$ assume the values 0 and $N+1$ respectively, and the variables $s_{\rm lo}$ and $s_{\rm hi}$ assume the empty line and the line consisting of several equal symbols with number 255. Thus, the condition $s_{\rm lo} < s_j < s_{\rm hi}$ for any line $s_j = S(j)$ from the list is provided.

Until then, either $A =$ False, or $J_{\rm hi} - J_{\rm lo} > 1$, the following actions are carried out:

The number $j = \mathcal{R}((J_{\rm lo}+J_{\rm hi})/2)$ and the line $s_j = S(j)$ are determined. Three cases are possible:

- At $s_j = s$, the variable A assumes the value True (thus further search for the number of the line stops);
- At $s_j < s$ the variable $J_{\rm lo}$ assumes the value of j and the variable $s_{\rm lo}$ – the value of s;
- At $s_j > s$ the variable $J_{\rm hi}$ assumes the value of j and the variable $s_{\rm hi}$ – the value of s.

If the line s in the considered list exists then the variable j will prove to be equal to the number of this line in the list; otherwise j will prove to be equal to the number of the line from the list most "similar" to s.

2. Files allowing for finding the necessary line from the list. For a practical realization of the algorithm described in the previous Item it is also necessary to have the possibility of a fast determination of the values of the functions $S(j)$ and $\widetilde{S}(j)$, i.e. the determination of the lines with the given numbers from the lists of the initial and translated lines.

One of the methods for solving this problem consists in the following.

The whole text of the dictionary should be contained in one text file; moreover, if N is the quantity of translated words and phrases then for each value $j = 0, ..., N-1$

- The line of the file-dictionary with number $3j+1$ is $S(j+1)$;
- The line of the file-dictionary with number $3j+2$ is $\widetilde{S}(j+1)$;
- The line of the file-dictionary with number $3j+3$ is not used and can contain any comments.

It is known that the text file in which any coding of the system Windows is used represents a sequence of bytes, i.e. integers from the diapason $(0, ..., 255)$, moreover each byte from the diapason $(32, ..., 255)$ corresponds to one symbol used for graphical representation of the text; other bytes correspond to the so-called operating symbols, in particular, some of the operating symbols are dividers of adjacent lines. In order that the written program was independent of the platform it is expedient to consider that the divider of lines is one symbol with number 10 (as it is accepted in the system Linux). If, as it is accepted in the system Windows, the divider of lines is a pair of symbols with numbers 13 and 10, then in the course of determining the line with the given number the excess symbol with number 13 is removed together with other excess symbols at the end of the line as a result of performance of the finishing operation over the line (see below).

Let us designate by r_j, for each value $j = 1, ..., N$, the number of the byte in the file-dictionary with which the line $S(j)$ begins. We will assign the natural number L_{bond} which is the upper boundary of the quantity of symbols for the majority of lines of the file-dictionary (in practice, it is most convenient to choose this number from the range $(128, ..., 256)$).

The line $s = S(j)$ is determined as follows:

1) In the beginning this line is assumed an empty string.

2) The file position moves to the byte with number r_j, and the sequence s' consisting of L_{bond} symbols is read from the file. This sequence is considered to be a text line. In it, the position K of the byte-divider of lines is determined.

3) If this byte-divider is found then the leading $K-1$ symbols of the string s' are "concatenated" to the right side of the string s, the last spaces and operating symbols are removed from the obtained string s (in the language Object Pascal the standard function "TrimRight" may be used for this purpose; in the language Java the user should independently write such function or use the standard function "trim" which deletes leading and trailing whitespace from strings, if only it is known that any of the

translated lines and the results of translations do not start with the whitespace). As a result, the obtained string s appears to equal the required line $S(j)$.

If this byte-divider is not found then the whole string s' is "concatenated" to the right side of the string s, the variable r_j assumes value $r_j + L_{\text{bond}}$, and return to the step 2) is carried out.

The string $s = \widetilde{S}(j)$ is determined as follows: at first the string $s_0 = S(j)$ is read from the file by the method described above, then the following string which is equal to required value s is similarly read.

Numbers r_j can be determined by means of the formula

$$r_{j+1} = \sum_{k=1}^{3j}(L_k + 1) \qquad (j = 0, ..., N-1),$$

where L_k is the quantity of symbols in the k th line of the text file-dictionary, to which the unnecessary deleted symbols before the symbol-divider of lines are added at the end (if they exist). The sequence of integers $[r_1, ..., r_N]$ is kept in an additional binary file. At each change of the text file-dictionary, there is a necessary "adjustment" of the electronic dictionary, which should include the determination of the sequence of the numbers $[r_j]$ and its saving in the additional file.

3. Realization of the algorithm. The methods and algorithms described here have been successfully used not only in the program introduced in the present work, but also in the earlier created packages of applied programs [5, 6, 7, 8, 9, 10, 11]. In these packages there is a choice of the working language, i.e. the language in which in the process of the package work text messages, headings of the main menu, and headings of auxiliary windows of the program are displayed. Thus, separate words or sentences are automatically translated from English by means of the available file-dictionary, and the user can create a dictionary for the translation of words and phrases in any language that is interesting for him or her.

The quantity of translated words and sentences in such packages amounts to anywhere from several hundred to several thousand. Thus, even while working with old computers with a 386 th processor, the delay in the appearance of text messages on the screen caused by the translation of corresponding lines is completely not appreciable.

5 Description of the available loaded files-dictionaries

Now there are following files-dictionaries intended for working with the described program:

1. Russian-English Dictionary of the Mathematical Sciences. Author: A.J.Lohwater. The number of phrases: 13313

2. Physics & Mathematics. English-Russian / Russian-English dictionary. Authors: A.G.Alenitsyn, E.I.Butikov. The number of phrases: 4028 / 4330

3. Physics & Mathematics. German-Russian / Russian-German dictionary. Authors: A.G.Alenitsyn, E.I.Butikov. The number of phrases: 4492 / 4330

4. Names often to be found in mathematical texts. English-Russian / Russian-English dictionary. The number of phrases: 1088 / 1139

5. Explanatory mathematical dictionary; Main terms (in Russian). Authors: A.M.Mikisha, V.B.Orlov. Printing edition: Moscow, "Russkiy yazik", 1989, 240 p. The number of phrases: 2335

6. Explanatory physical dictionary; Main terms (in Russian). Authors: A.V.Brukhanov, G.E.Pustovalov, V.I.Ridnik. Printing edition: Moscow, "Russkiy yazik", 1988, 232 p. The number of phrases: 3661

7. Big psychological dictionary. Authors: B.G.Mescheriakov, V.P.Zinchenko. 3nd ed., 2002. The number of phrases: 2353

8. General psychology. Author: A.V.Petrovskii (editor). Printing edition: Moscow, "PER SE", 2005, 251 p. The number of phrases: 439

9. The latest philosophical dictionary. Author: A.A.Gritsanov (editor). 3nd ed., 2003, 1782 p. The number of phrases: 955

10. The latest dictionary of political science. Authors: A.A.Pogoreliy, V.Yu.Fesenko, K.V.Filipov (editors). Printing edition: Rostov na Donu, "Feniks", 2010, 318 p. The number of phrases: 554

11. Dictionary of financial terms. The number of phrases: 2082

12. Short dictionary of literary terms. Authors: L.I.Timofeev, S.V.Turaev. Printed edition: Moscow, "Prosveschenie", 1985. The number of phrases: 519

13. Musical dictionary in stories. Author: L.V.Mikheeva. Printed edition: Moscow, "Sovetskiy kompozitor", 1984. The number of phrases: 265

14. Sports dictionary. // The number of phrases: 3662

15. Explanatory Russian dictionary. Authors: S.I.Ozhogov and N.Yu.Shvedova. †The number of phrases: 40607

16. Explanatory Russian dictionary. Author: D.N.Ushakov (editor). The number of phrases: 86659

17. Foreign words dictionary. †The number of phrases: 16105

18. Dictionary of the Russian synonyms and similar in sense expressions. Author: N.Abramov. Printed edition: Moscow, "Russian dictionaries", 1999. The number of phrases: 19743

19. English-Russian dictionary. Author: V.K.Müller (editor). Printing edition: Moscow, "Russian", 1995; Edition 24 corrected. The number of phrases: 66170

20. List of Christian names from the English-Russian dictionary. Author: V.K.Müller. The number of phrases: 642

21. English-Russian dictionary. Irregular English verbs. The number of phrases: 171

22. English-Russian / Russian-English dictionary. Author: N.Korolew. †The number of phrases: 33136 / 32363

23. English-Russian / Russian-English dictionary of the general lexicon. †The number of phrases: 17172 / 21289

24. English-Georgian / Georgian-English big dictionary. The number of phrases: 60499 / 49620

25. English-Georgian / Georgian-English small dictionary. The number of phrases: 8839 / 9596

26. English-Armenian / Armenian-English dictionary. Authors: Vardan Papoian and Vahan Papoian. The number of phrases: 7133 / 9546

In this list the dictionaries the text of which contains typos, but it is possible to work with them, are marked by the symbol "†".

Also, creation of the following dictionaries connected to the program is planned in the nearest future:

1. Explanatory dictionary of art (Russian–Georgian). Author: A.Kipshidze. Printing edition: Tbilisi, "Ganatleba", 1985.

2. Russian-Georgian / Georgian-Russian dictionary of the general lexicon.

6 Conclusion

The computer program for working with electronic dictionaries is created, to which several concrete dictionaries in the form of text and binary files are attached. The user has an opportunity to connect any of the available dictionaries to the program. The files presenting such dictionaries can be easily edited or created "from the beginning".

The original methods and algorithms developed by the author and providing high speed of work and stability results are used in the program. The program is realized (at this stage) for the operating system Windows according to the Standards for similar production, and the consumer can use it as a modern, convenient, simple and reliable tool. Long experimental investigations of the program and the algorithms realized in it have confirmed their high computing, operational and service qualities.

References

1. http://www.lingvo.ru/

2. https://en.oxforddictionaries.com/

3. Kachiashvili K.J., Melikdzhanian D.Yu., Prangishvili A.I. *Computing Algorithms for Solutions of Problems in Applied Mathematics and Their Standard Program Realization. Part 2 – Stochastic Mathematics.* Nova Science Publishers, Inc., New York; 2015.

4. Samarskii A.A., Gulin A.V. *Numerical methods.* (in Russian). Nauka, Moscow, 1989.

5. Kachiashvili K.J., Gordeziani D.G., Melikdzhanian D.Yu., Stepanishvili V.A. Packages of the applied programs for the solution of problems of ecology and processing of the experimental data. *Reports of Enlarged Sessions of the Seminar of I. Vekua Institute of Applied Mathematics*, **17**, 3 (200)2, 97–100.

6. Kachiashvili K.J., Gordeziani D.G., Lazarov R.G., Melikdzhanian D.I. Modeling and Simulation of Polutants Transport in Rivers. *International Journal of Applied Mathematical Modelling (AMM)*, **31**, (2007), 1371–1396.

7. Kachiashvili K.J., Melikdzhanian D.Yu. Software for Determination of Biological Age. *Current Bioinformatics*, **4** (2009), 41–47.

8. Kachiashvili K.J., Melikdzhanian D.I. Software Realization Problems of Mathematical Models of Pollutants Transport in Rivers. *Advances in Engineering Software*, **40**, (2009), 1063–1073.

9. Kachiashvili K.J., Melikdzhanian D.I. SDpro – The Software Package for Statistical Processing of Experimental Information. *International Journal of Information Technology & Decision Making (IJITDM)* **9**, 1 (2010), 115–144.

10. Kachiashvili K.J., Melikdzhanian D.I. Modern Software for the Environmental Modeling and Statistical Data Analysis. *Procedia Computer Science, WCIT-2010*, **3**, (2011), 439–443.

11. Kachiashvili K.J., Melikdzhanian D.I. *Advanced Modeling and Computer Technologies for Fluvial Water Quality Research and Control.* NOVA Scientific Publishers, Inc., 2012.