ISSN 2278-3083

Volume 2, No.4, July – August 2013

International Journal of Science and Applied Information Technology

Available Online at http://warse.org/pdfs/ijsait03242013.pdf

Software Implementation of Real Number Conversion into Basic Positional Notations with Controlled Accuracy

A. Anopriyenko [1], S. Ivanitsa [2], S. Kulibaba [3]

[1] PhD, professor, Donetsk National Technical University (DonNTU). Ukraine, anoprien@cs.dgtu.donetsk.ua

[2] PhD student, Donetsk National Technical University (DonNTU). Ukraine, isv@cs.dgtu.donetsk.ua

[3] MSc, Donetsk National Technical University (DonNTU). Ukraine

ABSTRACT

The present article concerns itself with the description of real numbers converter into basic positional notations (binary, denary, hexadecimal) with the controlled accuracy of fractional part of converted number formation. Here the converter functionality and the peculiarities of implementation of the used algorithms of converting long numbers from one numerical notation into the other without making use of the processor input/output are specified. Moreover, the analysis of the program action period while converting the numbers of different exponents has been carried out.

Keywords: Numerical notation, Real number conversion, Converter, High-precision arithmetic, Controlled accuracy.

1. INTRODUCTION

The numbers and the numerical notations, which produce these numbers, underlie the modern electronic computer engineering. The numerical notation may be understood as a concrete number representation with the help of written characters. The numerical notation (abbreviation NN is often used) is, in fact, a symbolic method of number designation, and it also gives each number (and lots of numbers in whole) the unique form and reflects the algebraic and arithmetic structure of numbers.

The parameters of computer systems and hardware, first of all speed and reliability index, vary depending on the effectiveness of the numerical notation [1-2].

In computer engineering the use is made of positional notations, which are identified by the integral number β — base number (β > 1).

The numbers, expressed by binary NN (β = 2), are used in the computer operations performed by a processor: notation, reading, addition, etc. The numbers presented in hexadecimal NN (hexadecimal format, β = 16) are used in computer for memory cell addressing. The Indo-arabic denary numerical notation with the base of β = 10 is the most wide-spread and customary for a human. The integral number without the sign x_1 in numerical notation with the base β is presented in the form of the finite linear combination of the number exponents:

$$x_1 = a_{n-1}a_{n-2}...a_0 = \sum_{k=0}^{n-1} a_k \beta^k, \quad (1)$$

where a_k — the integral numbers (numeral), which meet the inequality $0 \le a_k < \beta$; n — digit count x_1; k — index (the number of digit).

Any integral number may be represented by the finite digit count n. The real number without the digit x_2 in numerical notation with the base β is also presented in the form of finite linear combination of the number exponents:

$$x_2 = b_{n-1}b_{n-2}...b_0, c_1c_2...c_m = \sum_{k=0}^{n-1} b_k \beta^k + \sum_{l=1}^{m} c_l \beta^{-l}, \quad (2)$$

where b_k, c_k — digits of integer and fractional part of the digit x_2, meeting the inequality $0 \le (b_k, c_k) < \beta$; n, m — digit counts of integer and fractional parts of x_2 with indexes k and l respectively.

However, not any real number may be represented as a finite digit count [3-7]. For example, lots of irrational numbers are represented by the infinite fraction and is often fixed in round figures in the definite digit. Such number as x_3 in numerical notation with the base β may be represented in the form of the finite linear combination of integer part orders and the infinite linear combination of orders of its fractional part:

$$x_3 = b_{n-1}b_{n-2}...b_0, c_1c_2c_3... = \sum_{k=0}^{n-1} b_k \beta^k + \sum_{l=1}^{\infty} c_l \beta^{-l}, \quad (3)$$

The appropriate regulations (algorithms) are used for conversion of the number from one positional notation into the other. In the general case such algorithm boils down to the following operations:

1. **Using the arithmetic of the new numerical notation**: while converting the number from the current numerical notation with the base β1 into the new numerical notation with the base β2, it is necessary to fix the expansion, foundation coefficient and the indexes of orders of numerical notation with the base β2, using the arithmetic of the numerical notation with the base β2 and make all the operations in this notation.
2. **Using the arithmetic of the current numerical notation**: while converting the number from the current numerical notation with the base β1 into the new numerical notation with the base β2, it is necessary:

- to divide the integer part of the number represented in NN with the base β_1, into the base of NN with the base β_2, marking out the excesses. The last, written in a reverse order, make the integer part of the number in NN with the base β_2;
- to multiply sequentially the fractional part of the number, written in NN with the base β_1, by the base of the new NN with the base β_2, marking out the integer parts, which make the notation of the fractional part in NN with the base β_2.

The process of converting the numbers of large digit capacity from one NN into the other appears to be rather labor-consuming for manual calculation. This entails the use of different software tools, which are called program-converters for such operations. Particularly, each high-accuracy computing engine (Mathcad, Mathematic, MATLAB, etc. [8-9].) is able to convert and do arithmetic of the numbers in different numerical notations.

Also when developing the software for high-accuracy computing systems, the programmer during debugging of a program code has to analyze the intermediate outcome presented in the form of the set of machine words. Since the machine word is in fact binary number, it often happens that human perception needs the binary number presented in a more compact hexadecimal format or in the form of the decimal number.

The main goals of the present research can be formulated as:

1. The program implementation of algorithms of inverse conversion of real numbers in 2, 10, 16 NN.
2. The research of opportunities of the created converter that makes use of arithmetic's for conversion of numbers, exceeding possibilities of the PC central processor.

The received results of the research and engineering were approved at the III international scientific-technical conference "Information Management Systems and Computer Monitoring" which has taken place in Donetsk National Technical University on 16–18th of April, 2012.

2. SOFTWARE IMPLEMENTATION

The idea of creation of this software product originates in the cycle of the researches confirming efficiency of conversion into post-binary formats of numbers with a floating point [5–7], and while working over these numbers the task was that of converting of decimal numbers into binary with the help of getting high-accuracy real numbers (up to 10 000 signs after the point).

Before getting down to creation of the own converter, the attempt was made to find the ready-made program meant for conversion of fractional numbers with the necessary accuracy. However there were no full-function converter among those that have been considered that was able to prescribe the required accuracy of the converted number. The majority of the considered programs (including the standard calculator in Windows 7) worked only with integers. From among those software products which after all had coped with converting of fractional numbers, it is possible to point out, such programs as: ".:DOS:. Conversion of numbers" [10]; "Fedchenko's Converter" [11] and web services: "Notation online" [12]; "Wolfram Alpha" [13].

However none of the listed software products was in conformity with qualifying standards of the accuracy of converting of the fractional part of a number. Besides, any of the converters stated above didn't take into account the peculiarities of real numbers, lying in the fact that the majority of such numbers can't be presented in various notations identically. Therefore there arose the necessity of taking into account the error of fractional numbers converting, for example, when converting from decimal notation into the binary one. For more flexible check of the result it is necessary to take account of possibility of the choice of quantity of digits after a point.

The key parameters of the considered converters are specified in Table 1 with the indication of possibility of the choice of converting accuracy — the number of significant digits m of fractional part of the converted number.

Table 1: The parameters of the considered converters with converting accuracy

Program-converter or web service	Programming language	Choice of converting accuracy	*m*
Calculator in Windows 7	C#/VB.NET	–	–
Notation online	PHP	Yes	8
Fedchenko's Converter	C++	No	12
.:DOS:. Conversion of numbers	Visual Basic	No	232
Wolfram Alpha	Mathematica	No	2066

It should be pointed out that the Wolfram Alpha web service possesses the maximum accuracy of fractional numbers conversion among the considered products. However the procedure of converting the number is not quite convenient, since because of the absence of the choice of converting accuracy the result is read out piece by piece and it is necessary to update the result each time to get a new portion. The result received by the Wolfram Alpha service is read out in the form of the graphic file and it is necessary to additionally apply means of character recognition to use it as text data.

Thus, the created bin-dec-hex fractional number converter (in abbreviated form bdh-converter) is deprived of the bugs that a number of analogs stated above have, and in fact is able to operate with real numbers of any length (the set limit in 10 000 signs after a point may be increased if necessary).

The bdh-converter program carries out interconversion into binary, decimal and hexadecimal NN. The converter is

implemented in the Java language and occupies about 200 Kb of disk space. The choice of the Java programming language is conditioned, first of all, by a cross-platform, the convenience of the description of model-based processes and the turbo speed [14-15]. The operating space of the bdh-converter program is presented in Figure 1.

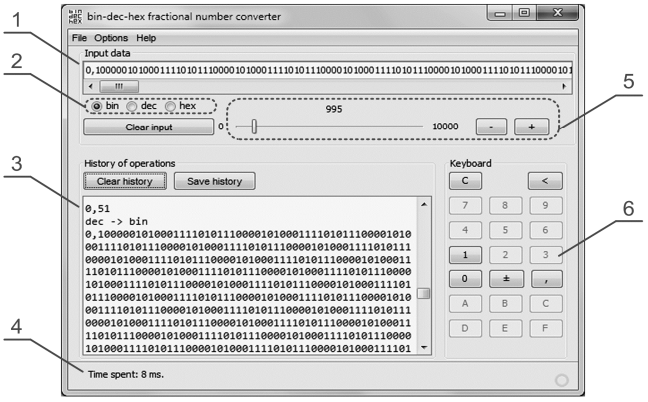


Figure 1: Operating space of bdh-converter with the indication of basic fields: 1 – entry field; 2 – the choice of numerical notation for conversion; 3 – history of operations; 4 – status line; 5 – the choice of output accuracy; 6 – interactive keyboard

An entry field (Fig. 1, field 1) is intended for number notation which needs to be converted into another notation. The input of initial number can be made with the help of a personal computer keyboard, a clipboard or by the instrumentality of the additional interactive (virtual) keyboard (Fig. 1, field 6). Along with this the input line undergoes the analysis of correctness of number input. In the engineering normalized notation the input of symbols in the chosen notation, the character of the entered number, the divider of fractional and integer parts, and also the character of the exponent *e* with the number input mark is permitted. If the input is made incorrectly, the entry line is marked out with a red color.

The selecting field between the notations (Fig. 1, field 2) is dominant for the choice of the converting direction. While choosing one of the selectors in NN, there is a conversion of initial number into the chosen system, with regard for the fixed error which is specified by the quantity of the significant characters after a point. The conversion being implemented, the chosen NN becomes current.

The number of significant digits after a point is fixed with the help of the scroll box and the "+" and "–" buttons (Fig. 1, field 5). The current value of accuracy is displayed over the scroll box, and to its left and to the right boundary values of accuracy (in the current implementation $0 \div 10\,000$) are indicated. The reading of the scroll box position happens only during the conversion of input data.

The conversion into the new NN being made, the conversion result replaces the initial number in the entry field, and is also logged in the history (Fig. 1, field 3). After the switching of the numerical notations there happens the activity change of buttons of the virtual keyboard (only those keys which correspond with the current numerical notation are always accessible).

The sequence of actions carried out by the user is fixed in the history of converting operations in the format: "initial number, the direction of conversion, the result and converting time". Such format of fixation makes it possible to observe all the operations performed by the user. For example, in the history the following information is reflected in figure 1: the user entered decimal number 0.51 and got its binary equivalent with an accuracy of 1 000 digits after a point.

The "Clear the history" button carries out data cleaning in the history field, and the "Save the History" button aims at saving the history of converting operations of converting in a text file.

In the status line (Fig. 1, field 4) the time of the last converting in milliseconds, and also the pictogram of activity peculiar of the majority of Java-applications are displayed.

The existence of the multilingual interface, such as Russian, Ukrainian, English and German, is also a distinctive feature of bdh-converter. During the application start the language of the interface is being fixed automatically, depending on the chosen locale in the operating system of the user. Later on the language may be changed in menu item "Options".

Thus, the following features may be singled out as the peculiarities of the bdh-converter:

1. **Cross-platform**. This program is being coded in the Java language, which virtual machine enables it to start applications both in Windows OS, and in *nix-like systems.
2. **User-friendly interface**. Due to the growth of tablet computers and touch screens, the presence of the keyboard display is hailed, because the standard keyboard display is not always practical. Also, the main functions of the program are duplicated in combinations of keys for a fast call.
3. **Controlled accuracy of conversion**.
4. **Keeping the history of converting** with the possibility of saving in a file.
5. **Loading of the initial data from the file.** It is not always convenient to enter long numbers by hand, especially, when they are the results of preceding calculations. In some cases such data are stored as a text in a file. The program may input the text into the number entry field.
6. **Multilanguage interface.**

The peculiarity of bdh-converter, unlike similar converters, stretches further its functionality and concerns also an algorithmic base. First of all, the program does not use mathematical capability of the PC central processor since any of today's PC processors isn't able to do so accurate calculations.

The program presents all the data in the form of character string, and makes the necessary conversions, using the mechanisms of character-stepped processing. Such procedures implement program mathematics (i.e. all the mathematical operations are performed as the text string operations), which speed of calculation is lower than that of calculations with hardware support.

However during the work even with the longest numbers, the waiting period doesn't put the program into the forced closedown mode. The block diagram of algorithm of the bdh-converter is shown in Fig. 2, 3.

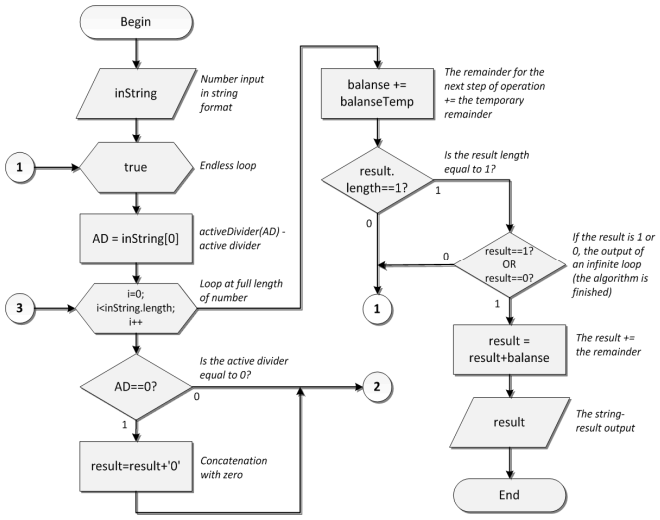


Figure 2: The block diagram of algorithm of the bdh-converter (the beginning)

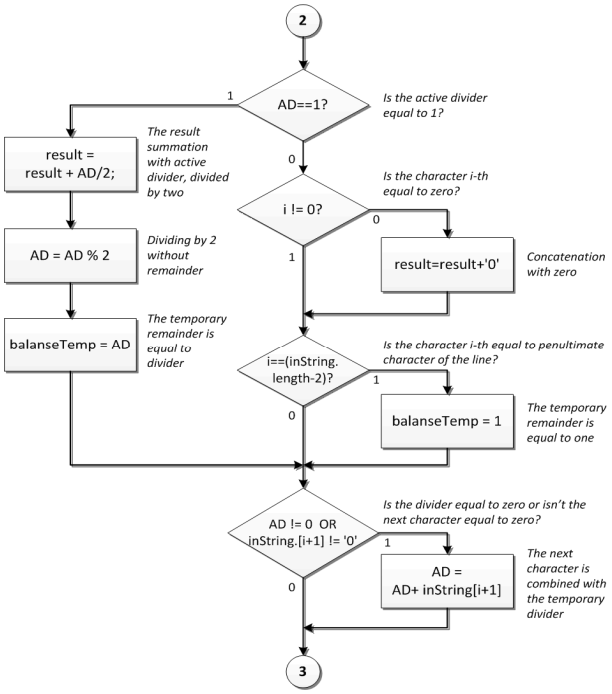


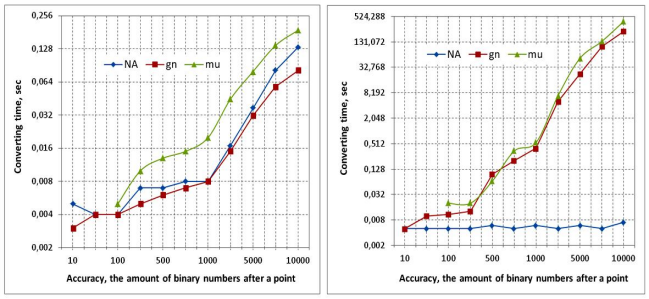
Figure 3: The block diagram of algorithm of the bdh-converter (the end)

The conversion of numbers of different orders from 10 NN into 2 NN in the whole radius of accuracy of the result presentation was being made to assess the program speed. As the test numbers the approximate values of physical constants were taken [16]: $N_A = 6.0221407818 \cdot 10^{+23}$ mole^{-1} — Avogadro constant; $g_n = 9.80665$ м·с$^{-2}$ — the standard acceleration of a free fall on the Earth surface; $m_e = 9.10938215 \cdot 10^{-31}$ kg — electron mass.The results of operating time of the program are presented in Fig. 4*a*.

It should be pointed out that the initial decimal numbers of different orders were converted into binary numbers of the maximum accuracy (10 000 characters after a point) with the values of time investment which aren't exceeding 0.25 sec.

The results of program operating time during back conversion of the received binary values of test numbers into the initial 10-mal notation are shown in Fig. 4*b*.

The converting time has considerably increased because the input binary number has the same accuracy of representation, as well as the expected decimal number. However, Avogadro constant, owing to a positive exponent, is represented by the program as an integral number (the algorithm of integers conversion is involved only), and the time of its conversion in the whole radius of accuracy is identical: 5±2 ms. It should be noted that another two binary numbers of different orders were converted into the initial decimal numbers with the maximum accuracy with the values of time investment which weren't exceeding 9 min.



a) *b)*

Figure 4: The converting time of test numbers – accuracy of the result representation diagram (*a* – from 10 NN into 2 NN; *b* – from 2 NN into 10 NN)

3. CONCLUSION

Thus, the introduced bdh-converter is a good means of getting numbers in various numerical notations nowadays. The topicality of the program, except the described differences, may be expressed by two more important aspects:

1. During the software development for high-accuracy computing systems, a programmer needs software tools for the analysis of the intermediate outcome. The

programming tool set should have a program, which is able to convert the numbers into the various numerical notations.

2. Recently the growth of number of experts in computer science and technologies is observed. Consequently, the increasing number of students of computer profession in institutes of higher education begin studying from the basis of informatics, there they face the representation of real number in various notations. In such a case the bdh-converter program may be not only the working tool for programmers or engineers of computer systems, but the trainer for students as well.

REFERENCES

1. Anopriyenko A. **Tetralogic and tetracodes: an effective method for information coding.** 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics. Berlin, August 24-29, 1997. Vol. 4. Artificial Intelligence and Computer Science. – Berlin: Wissenschaft und Technik Verlag. – 1997, p. 751-754.
2. Anopriyenko A., Svjatnyi V., Reuter A. **Extended logical and numerical basis for computer simulation**. Short Papers Proceedings of the 1997 European Simulation Multiconference ESM'97. Istanbul, June 1-4, 1997 – Istanbul, SCS, 1997, p. 21-22.
3. Bailey D.H. **High-Precision Arithmetic in Scientic Computation**. Computing in Science and Engineering, May-Jun, 2005, p. 54-61, http://crd.lbl.gov/ dhbailey/dhbpapers/high-prec-arith.pdf.
4. Bailey D.H., Borwein J.M., **High-Precision Numerical Integration: Progress and Challenges**. July 14, 2009, http://www.davidhbailey.com/dhbpapers/hp-num-int.pdf
5. Bailey D.H. **Resolving numerical anomalies in scientific computation**, manuscript, Jan 2008. http://www.davidhbailey.com/dhbpapers/numerical-bugs .pdf.
6. Bailey D.H., Jonathan M.B. **High-precision computation and mathematical physics**. XII Advanced Computing and Analysis Techniques in Physics Research, 2008, http://www.davidhbailey.com/dhbpapers/dhb-jmb-acat08 .pdf.
7. Bailey D. H., Barrio R. and Borwein J. M. **High precision computation: Mathematical physics and dynamics.** Applied Mathematics and Computation, vol. 218 (2012), p. 10106-10121.
8. **High-Precision Software Directory**. http://crd-legacy.lbl.gov/~dhbailey/mpdist/.
9. Bailey D.H., Yozo Hida, Xiaoye S. Li and Brandon Thompson, **ARPREC: An Arbitrary Precision Computation Package**, 2002, http://crd.lbl.gov/~dhbailey/dhbpapers/arprec.pdf.
10. **The program to convert numbers from one number system to another with breeding translation step by step**. http://pascalstudy.narod.ru/loading/opred.zip.
11. **Fedchenko's Converter**. http://instrument-p.narod.ru/_progr/konvert.htm.
12. **Translation numbers in binary, hexadecimal, decimal, octal number system online**. http://math.semestr.ru/inf/index.php.
13. **Wolfram Alpha: Computational Knowledge**. http://www.wolframalpha.com.
14. Kahan W. and Darcy J. **How Java's Floating-Point Hurts Everyone Everywhere**, available at http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf, 1998.
15. David H.B. **Java meets numerical analysis**, Scientific Programming, vol. 12 (2004), no. 1, pg. 59-60.
16. **Fundamental Physical Constants — Complete Listing**. http://physics.nist.gov/cuu/Constants/Table/allascii.txt.