UDC 004.021
Doi: 10.31772/2587-6066-2019-20-2-191-196

For citation: Udalova J. V., Kuzmin D. A. [Library of mathematical functions with parallelism at the operational level in the Pythagor language] *Siberian Journal of Science and Technology.* 2019, Vol. 20, No. 2, P. 191–196. Doi: 10.31772/2587-6066-2019-20-2-191-196

Для цитирования: Удалова Ю. В., Кузьмин Д. А. Реализация библиотеки математических функций с параллелизмом на уровне операций на языке Пифагор // Сибирский журнал науки и технологий. 2019. Т. 20, № 2. С. 191–196. Doi: 10.31772/2587-6066-2019-20-2-191-196

LIBRARY OF MATHEMATICAL FUNCTIONS WITH PARALLELISM AT THE OPERATIONAL LEVEL IN THE PYTHAGOR LANGUAGE

J. V. Udalova*, D. A. Kuzmin

Siberian Federal University
79, Svobodny Av., Krasnoyarsk, 660041, Russian Federation
*E-mail: judalova@sfu-kras.ru

At present, developed tools and libraries have been designed for imperative and functional programming languages that provide parallelism through processes or threads. There are other alternative approaches to the organization of parallel computing, one of which is implemented in Pythagor – the language of functional-streaming parallel programming, and involves parallelism at the level of operations.

The tools of the Pythagor programming language are actively developing, and the repository of predefined functions is expanding. Many mathematical functions have been designed to provide a developer with no less functionality than the math library math.h of the C programming language. A large part of the mathematical functions have been implemented using the Maclaurin's series. It is both used as an approach of faster and less accurate calculations, in which a predetermined number of elements of the series is calculated without cycles and recursions with the substitution of pre-calculated coefficients in the function code, and as an approach of less rapid and more accurate calculations, in which the elements of the series are calculated dynamically until the desired accuracy is achieved.

The development of a library of mathematical functions of a programming language is an applied algorithmic task already implemented in one way or another for a number of existing programming languages. But in many languages, the implementation of algorithms for mathematical functions is hidden from the user, while modern tools of the Pythagor language support an open repository of functions. Additional interest is the possibility of parallelism at the level of operations in the calculation of mathematical formulas in the Pythagor language.

Keywords: parallelism at the operation levels, functional-stream programming, algorithms of mathematical functions.

РЕАЛИЗАЦИЯ БИБЛИОТЕКИ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ С ПАРАЛЛЕЛИЗМОМ НА УРОВНЕ ОПЕРАЦИЙ НА ЯЗЫКЕ ПИФАГОР

Ю. В. Удалова*, Д. А. Кузьмин

Сибирский федеральный университет
Российская Федерация, 660041, г. Красноярск, просп. Свободный, 79
*E-mail: judalova@sfu-kras.ru

К настоящему времени параллельное программирование обеспечивается большим объемом развитых инструментов и библиотек, базирующихся на императивном программировании с применением параллельных процессов или потоков (нитей), также развиваются средства распараллеливания и для функциональных языков программирования. Вместе с перечисленными инструментами существуют и альтернативные подходы к организации параллельных вычислений, один из которых реализуется языком функционально-потокового параллельного программирования Пифагор, поддерживающим параллелизм на уровне операций.

И теоретические концепции, и инструментальные средства обозначенного языка программирования активно развиваются, расширяется репозиторий разработанных функций. Разработано множество математических функций, без встроенной реализации которых затруднено комфортное программирование многих задач, способное предоставить разработчику не меньшую функциональность, чем математическая библиотека math.h языка С. Большая часть математических функций реализована с помощью рядов Маклорена. Используется как подход, предоставляющий более быстрые и менее точные вычисления, при котором без циклов и рекурсий вычисляется предопределенное количество элементов ряда с подстановкой в код функции заранее вычисленных коэффициентов, так и подход, предоставляющий менее быстрые и более точные вычисления, при котором элементы ряда вычисляются динамически до достижения нужной точности. Для части функций ряд Маклорена имеет ощутимо разный уровень точности в рамках своей области определения, тогда в окре-

стностях точек, отрицательно влияющих на точность ряда, искомая функция уточняется с помощью дополнительных математических формул, например, формул приведения.

Задача описания библиотеки математических функций языка является прикладной алгоритмической задачей, уже реализованной тем или иным образом для ряда существующих языков программирования. При этом во многих языках реализация алгоритмов математических функций скрыта от пользователя, последнему предоставляется только возможность программного вызова такой функции, тогда как современные инструментальные средства языка Пифагор поддерживают открытый репозиторий функций. Применительно к языку программирования Пифагор интерес представляют особенности и возможности распараллеливания на уровне операций при вычислении математических формул, представленные в статье.

Ключевые слова: параллелизм на уровне операций, функционально-потоковое программирование, алгоритмы математических функций.

Introduction. The Pythagor language of functional-stream parallel programming [1–6] maintaining overlapping at the level of operations and assuming the organization of architecture-independent parallel calculations represents alternative approach to parallel calculations. Now the language and its tools are actively developing [7–14]. The article is devoted to realization of mathematical functions library for the Pythagor language and illustrates features of stream parallel calculations with overlapping at the level of operations.

Considerable part of mathematical functions is calculated by means of Makloren's series [15]. Both a faster and less accurate calculations, at which the predetermined number of elements of a series with pre-calculated coefficients, and a less fast and more accurate calculations, at which elements of series are calculated dynamically before achievement of the desired accuracy, are used.

Approximate calculation of mathematical functions and their program realization in a number of the known programming languages are the tasks that have been solved by now. As for the Pythagor language and its original calculation pattern the task has been solved for the first time. Besides, realization of mathematical functions in many known languages is hidden from the developer: he can cause function performance, but cannot know by means of what method and algorithm the calculation of result is done. Thus, the basic principles of mathematical library design described in the article can be useful to those readers who need to solve a similar problem for their own calculation system.

Fast approximate calculation of Makloren's series on the example of a sine, a cosine and exponent. Makloren's series for the sine is written as:

$$\sin(x) \approx x - x^3/3! + x^5/5! - x^7/7! + \ldots,$$
$$\text{where } |x| \le 1. \qquad (1)$$

In calculation of a sine for a random angle = x radian, it is required to give an argument to an interval $[-\pi, \pi]$. In case the argument belongs to $[-\pi, -\pi/2) \cup (\pi/2, \pi]$, it moves to the interval $[-\pi/2, \pi/2]$, whereupon remembered is the indicator of the subsequent multiplication of the result by –1. For the argument x from $[-\pi/2, \pi/2]$ calculated is $x' = 2x/\pi$, approximate formula (1) is written as follows:

$$\sin(x) = \sin(\pi x'/2) \approx (\pi/2)x' - (\pi/2)^3 x'^3/3! + (\pi/2)^5 x'^5/5! - (\pi/2)^7 x'^7/7! + \ldots,$$
$$\text{where } |x| \le \pi/2,\ |x'| \le 1 \qquad (2)$$

For fast calculation of an acceptable result the first six summands in a series are taken and their numerical coefficients are pre-calculated external to the program code: 1) $k1 = \pi/2 = 1.570796326794$, 2) $k2 = -(\pi/2)^3/3! = -0.645964097505$, 3) $k3 = (\pi/2)^5/5! = 0.079692626245$, 4) $k4 = -(\pi/2)^7/7! = -0.0046817541$, 5) $k5 = (\pi/2)^9/9! = 0.00016044118$, 6) $k6 = -(\pi/2)^{11}/11! = -0.0000035988432$.

In the course of the function operation, argument degrees are calculated; multiplication by coefficients and summation of a required formula are performed. Such algorithm is comparable to a similar consecutive imperative algorithm except the fact that the Pythagor language calculation pattern assumes potential parallel performance of all those operations relating to one or different functions with the available input data. Realization of the designated algorithm by means of the Pythagor language forms the following sequences of operators potentially available for parallel calculation (fig. 1).

Fig. 1 shows that a great number of operators of one branch, for example, calculation of argument degrees x2, x3, x5, etc. can be performed only consistently since they need the results of smaller degrees, whereas operators of one layer, for example, «(x2, x5):*>>x7», «(0.079692626245,x5):*>>p3», «(p1,p2):+>>S1» can be potentially available for parallel performance.

When illustrating the graph in fig. 1 selection of auxiliary identifiers was generally used – x2, … x11, p1, … p6, S1, … S5. The program code of function can be realized in the same style, or by means of combining formulas in one expression. For example, calculation of the required sum in the code is written as «(((((p1,p2):+,p3):+,p4):+,p5):+,p6):+ >> return», whereby the operators calculation pattern remains the same (fig. 1). But, for example, selection in the code of identifier x2 is desirable, its existence prevents performing operation 'x*x' several times when calculating argument degrees.

Comparison of results of a sine fast approximate calculation in the Pythagor language and calculations of a sine by means of a sin function call from math.h library in language C is presented in tab. 1.

Other trigonometrical functions can be calculated similarly via their Makloren's series or by means of substitution of sine values in reduction formulas, for example, of $\cos(x) = 1 - 2\sin^2(x/2)$.

When realizing the mathematical functions library for the Pythagor language, reduction formulas were used. Therefore tables of comparison of cosine and a tangent calculations in the Pythagor and C languages accurately correlate with tab. 1, i. e. for example, cos(1.2) will be closer to a similar call of a cosine from math.h than cos(1.6).

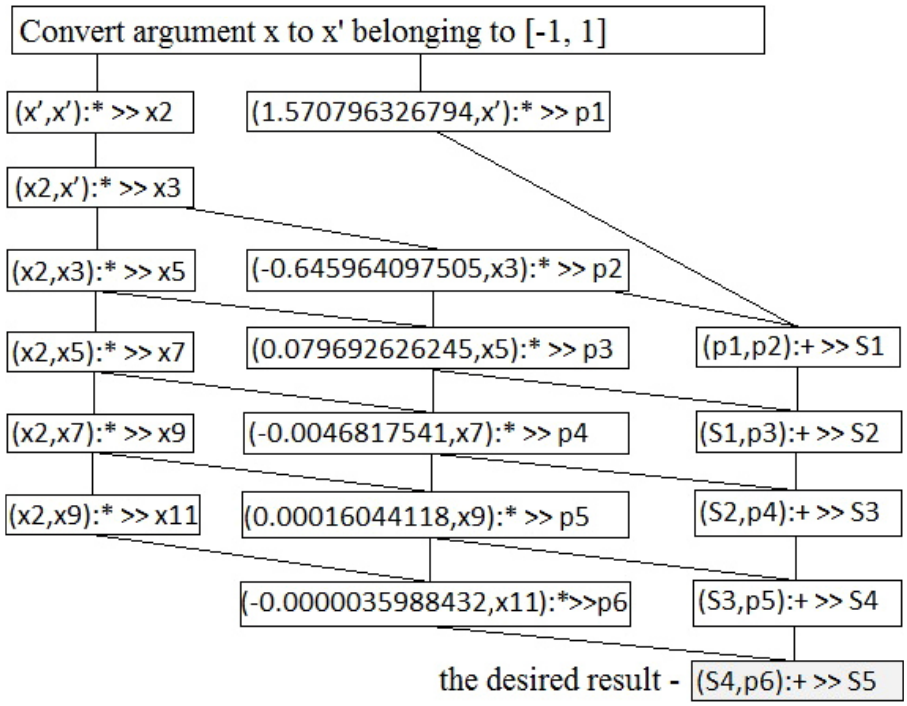

Fig. 1. Simplified information and control graph of the function sin

Рис. 1. Упрощенный информационно-управляющий граф функции sin

Table 1

Sine calculation in the Pythagor language and C language (library math.h)

Argument x	math.h	Pythagor	Argument x	math.h	Pythagor
0.2	0.198668	0.198669	5.2	–0.883455	–0.883455
0.4	0.389417	0.389418	5.4	–0.772766	–0.772764
0.6	0.564641	0.564642	5.6	–0.631268	–0.631266
0.8	0.717355	0.717356	5.8	–0.464604	–0.464602
1.0	0.841470	0.841471	6.0	–0.279418	–0.279416
1.2	0.932039	0.932039	10.0	–0.544021	–0.544021
1.4	0.985450	0.985452	50.0	–0.262375	–0.262375
1.6	0.999574	0.999578	100.0	–0.506366	–0.506370

Under approximate calculation of given functions by means of Makloren's series a considerable decrease in accuracy in the neighborhood of separate points, lying in the series range of definition can be observed, as a rule, it is 1 and-1. The arcsine belongs to such functions. For an arcsine the range of definition of function and series coincide, it is [-1, 1] therefore argument transformation, similar to the one performed to the sine, is not required. Under approximate calculation of an arcsine the following formula based on its Makloren's series with the coefficients calculated out of the program code is used:

$$\arcsin(x) \approx x + 0.166666666666x^3 + 0.075x^5 + \\ + 0.044642857142x^7 + 0.030381944444x^9 + \\ + 0.022372159090x^{11} + 0.017352764423x^{13} + \\ + 0.01396484375x^{15} + 0.011551800896x^{17} + \\ + 0.009666219208x^{19} + 0.008390335809x^{21} + \\ + 0.007312525873x^{23} + 0.000011679728x^{25}, \\ \text{where } |x| \le 1 \qquad (3)$$

While for a sine acceptable accuracy it is enough to take first six summands, for an arcsine it is desirable to calculate not less than twelve. In addition, supposing x is located in the neighborhood of points 1 or –1, the accuracy of an approximate formula considerably decreases. In realization of an arcsine for the Pythagor language at x belonging to [–1, –0.93) U (0.93, 1], the auxiliary formula $\arcsin(x) = \pi/2 - \arcsin$ is used $((1 - x^2)^{1/2})$.

Comparison of the arcsine fast approximate calculation in the Pythagor language and calculations of the arcsine by means of the sin function call from math.h library in language C is presented in tab. 2.

When calculating the exponent the following formula with preliminary calculated coefficients based on Makloren's series is used:

$$e^x \approx 1 + x + 0.5x^2 + 0.166666666666\,x^3 + \\ + 0.041666666666\,x^4 + 0.008333333333x^5 + \\ + 0.001388888888x^6 + 0.00019841269841x^7, \\ \text{where } |x| \le 0.7 \qquad (4)$$

Makloren's series of exponent is defined on [–1, 1], but as at fast approximate calculation the final small number of elements is used, for acceptable accuracy the domain [–0.7, 0.7] is chosen. The domain of exponent

definition includes all numerical axis, therefore for random argument x the exponent property is used $e^{a+b} = e^a e^b$. The argument x is divided into the integral and fractional parts, e in fractional degree is calculated with the help of formula (4), e in the integral degree is calculated by multiplication of constants e or 1/e, at the same time additional values, calculated external the program code, are applied to speed up calculations, for example, e^2 = 7.389056098930649, e^8 = 2980.957987041726, e^{64} = 6.235149080811582e + 027, used in multiplication.

Comparison of results of fast approximate calculation of exponent in the Pythagor language and exponent calculations by means of an exp function call from math.h library in language C is presented in tab. 3.

The accurate recursive calculation of Makloren's series on the example of a logarithm. Makloren's series for the natural logarithm is written as:

$$\ln(x + 1) \approx x - x^2/2 + x^3/3 - x^4/4 + ..., \quad \text{where } -1 < x \le 1. \qquad (5)$$

Formula (5) includes higher coefficients than Makloren's series of the above considered functions, therefore fast calculation of the natural logarithm is not realized as it would demand enumeration of a significant number of summands (up to several hundreds or thousands). Instead, recursive calculation of formula (5) before achievement of the desired accuracy of 10-8 is used, in imperative language it could be realized by means of a cycle, in the Pythagor language cyclic algorithms are realized by means of recursive functions.

The definition domain of the natural logarithm is a set of positive numbers, therefore for any argument x the property of the natural logarithm is used $\ln(b*10^a) = \ln(b) + a*\ln(10) = \ln(b) + a*2.302585092994046$, where b can be chosen so as to suit the calculation of ln(b) with the help of formula (5).

Recursive function expects the argument type of (number 1, number 2, number 3) where number is b, number 2 in the first recursion call coincides with b, number 3 is a series denominator, in the first recursion call equals one. One iteration of the recursion calculates four summands of Makloren's series. The simplified information and control graph of the natural logarithm function is written as (fig. 2).

The expressions in braces, for example {(S, (b, (b4,b):*, (z,4):+):recursion):+} in fig. 2, belong to the delayed calculations and will be carried out only if the condition of stop is true: - i.e. when the fourth command s4 in the current iteration taken according to module are more than 10^{-8}. Commands (b, (b4,b):*,(z,4):+) form the argument (number1, number2, number3) for the iteration following the same recursive function. Presented here recursive calculation of formula (5) can be realized via Pythagor language tools or other recursion ways, however, when realizing the library of mathematical functions the suggested method was chosen.

Comparison of the natural logarithm in the Pythagor language and natural logarithm by means of a log function call from math.h library in language C is presented in tab. 4.

Table 2

Calculation of the arcsine in the Pythagor and C languages (library math.h)

Argument x	math.h	Pythagor	Argument x	math.h	Pythagor
0.1	0.100167	0.100167	0.91	1.143284	1.140812
0.2	0.201358	0.201358	0.92	1.168081	1.164584
0.3	0.304693	0.304693	0.93	1.194413	1.189442
0.4	0.411517	0.411517	0.94	1.222630	1.229622
0.5	0.523599	0.523599	0.95	1.253236	1.258546
0.6	0.643501	0.643501	0.96	1.287002	1.290796

Table 3

Exponent calculation in the Pythagor and C languages (library math.h)

Argument x	math.h	Pythagor	Argument x	math.h	Pythagor
–25	0.000000	0.000000	0.5	1.648721	1.648721
–13.7	0.000001	0.000001	1.1	3.004166	3.004166
–9.6	0.000068	0.000068	6.6	735.095093	735.094910
–7.9	0.000371	0.000371	8.8	6634.245117	6634.240234
–1.5	0.223130	0.223130	16.3	11994985.0	11994990.0
–0.8	0.449329	0.449329	25	72004902912	72004902912

Table 4

Calculation of natural logarithm in Pythagor and C languages

Argument x	math.h	Pythagor	Argument x	math.h	Pythagor
0.07	–2.659260	–2.659260	15	2.708050	2.708050
0.15	–1.897120	–1.897120	40	3.688879	3.688879
0.47	–0.755023	–0.755023	200	5.298317	5.298317
0.88	–0.127833	–0.127833	400	5.991465	5.991465
1.85	0.615186	0.615186	10500	9.259130	9.259131
8	2.079442	2.079442	100000	11.512925	11.512930

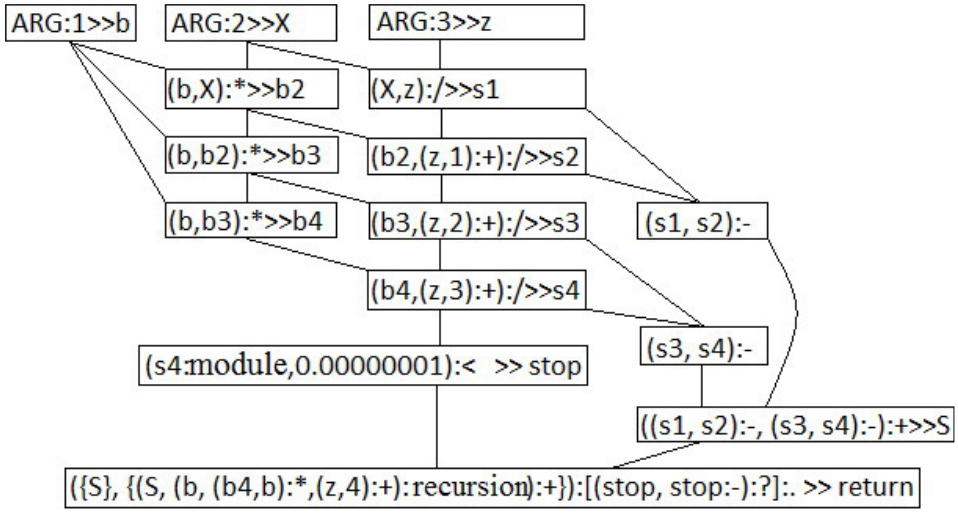

Рис. 2. Упрощенный информационно-управляющий граф функции ln

Fig. 2. Simplified information and control graph of the function ln

Having realization of natural logarithm calculation it becomes possible to calculate a logarithm on a random basis, using $\log_a$ (b) property = ln(b)/ln(a), for a decimal logarithm in a denominator a constant 2.302585092994046 calculated exclusive the code is substituted. Exponentiation is realized by means of the formula $x^y = e^{y*\ln(x)}$.

Conclusion. The problem of the Pythagor language mathematical library development has been completely realized and is one of the subtasks which are successfully executed with the RFFI grant "Architecture-independent Development of Parallel Programs on the basis of the Functional and Stream Paradigm".

The list of the realized mathematical functions. Module. Sine. Cosine. Tangent. Cotangent. Arcsine. Arccosine. Arctangent. Arc cotangent. Secant. Cosecant. Square root. Cubic root. Rounding. Rounding down. Rounding up. Integral and fractional parts of number. Exponent. Hyperbolic sine, cosine, tangent, cotangent. Natural logarithm. Decimal logarithm. A logarithm on the specified basis. Exponentiation. The integral part from division and a remainder of division (arguments of function can be numbers, both integral, and material). Mantissa and exponent of the two marking. Multiplication of number to the power of two. Number sign. Inverse hyperbolic: cosine, sine, tangent, cotangent, secant, cosecant. Rising of the two to the power. Maximum from couple of numbers. Minimum from couple of numbers. Hypotenuse. Difference module.

All realized functions are included in the open repository of the Pythagor language, thus, the developer can not only execute and look through their code, but also borrow it for creation of own algorithms or alternative realization of mathematical functions.

Acknowledgment. This work was carried out under financial support of RFFI within the framework of the scientific project № 17-01-00001.

Благодарности. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-01-00001.

References

1. Legalov A. I., Ushakova M. S. [Features of the development and transformation of functional data-flow parallel programs]. *Superkomp'yuternye dni v Rossii. Trudy mezhdunarodnoy konferentsii* [Supercomputer days in Russia. Proceedings of the international conference]. 2018, P. 999–1000.

2. Legalov A. I. et al. [Change of computing management strategies in architecture-independent parallel programming]. *Nauchnyy servis v seti Internet. Trudy XIX Vserossiyskoy nauchnoy konferentsii* [Scientific service on the Internet. Proceedings of the XIX all-Russian scientific conference]. 2017. P. 341–350 (In Russ.).

3. Legalov A. I. [Language support for architecture-independent parallel programming]. *Yazyki programmirovaniya i kompilyatory. Trudy konferentsii* [Programming languages and compilers. Proceedings of the conference]. 2017, P. 169–172 (In Russ.).

4. Legalov A. I. et al. [Technological aspects of creation, transformation and execution of functional-stream parallel programs]. *Nauchnyy servis v seti Internet: vse grani parallelizma. Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii* [Scientific service on the Internet: all facets of parallelism. Proceedings of the International supercomputer conference]. 2013, P. 443–447 (In Russ.).

5. Legalov A. I. et al. [Storage of functional data-flow parallel programs]. *Vestnik SibGAU*. 2013, No. 4 (50), P. 53–57 (In Russ.).

6. Legalov A. I. et al. [Event-driven computing model that supports the execution of functional-stream parallel programs] *Sistemy. Metody. Tekhnologii*. 2012, No. 1 (13), P. 113–119 (In Russ.).

7. Romanova D.S. [Organization of testing to check the correctness of Pythagor's tools]. *Prospekt Svobodnyy – 2018. Materialy Mezhdunarodnoy konferentsii molodykh uchenykh* [Prospect Free – 2018. Proceedings of the International conference of young scientists]. Siberian Federal University, 2018, P. 25–28 (In Russ.).

8. Udalova U. V. [Verification of functional-stream parallel programs using interval formulas] *Obra-*

zovatel'nye resursy i tekhnologii. 2016, No. 2 (14), P. 259–262 (In Russ.).

9. Ushakova M. S., Legalov A. I. [Verification of programs with mutual recursion in Pythagor]. *Modelirovanie i analiz informatsionnykh sistem*. 2018, Vol. 25, No. 4 (76), P. 358–381 (In Russ.).

10. Vasiliev V. S., Legalov A. I. [Optimization of loop invariant in the language of Pythagor]. *Modelirovanie i analiz informatsionnykh sistem*. 2018, Vol. 25, No. 4 (76), P. 347–357 (In Russ.).

11. Legalov A. I. et al. A toolkit for the development of data-driven functional parallel programmes. *Communications in Computer and Information Science*. 2018, Vol. 910, P. 16–30.

12. Legalov A. I. et al. [Tool support for the creation and transformation of functional data-flow parallel programs]. *Trudy Instituta sistemnogo programmirovaniya RAN* [Proceedings of Institute for system programming of the Russian Academy of Sciences]. 2017, Vol. 29, No. 5, P. 165–184 (In Russ.).

13. Ushakova M. S., Legalov A. I. [Instrumental support for formal verification of programs written in the language of functional-streaming parallel programming]. *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika.* 2015, Vol. 4, No. 2, P. 58–70 (In Russ.).

14. Udalova U. V., Legalov A. I. [Verification of functional-stream parallel programs by inductive assertions]. *Doklady Akademii nauk vysshey shkoly Rossiyskoy Federatsii*. 2014, No. 2-3 (23-24), P. 125–132 (In Russ.).

15. Lusternik L. A. *Matematicheskii analiz. Funkcii, predeli, rydi, chepnie drobi* [Mathematical analysis. Functions, limits, series, continued fractions]. Moscow, IL Publ., 2012, 442 p.

Библиографические ссылки

1. Легалов А. И., Ушакова М. С. Особенности разработки и преобразования функционально-потоковых параллельных программ // Суперкомпьютерные дни в России. Тр. Междунар. конф. 2018. С. 999–1000.

2. Изменение стратегий управления вычислениями при архитектурно-независимом параллельном программировании / А. И. Легалов [и др.] // Научный сервис в сети Интернет : тр. XIX Всеросс. науч. конф. 2017. С. 341–350.

3. Легалов А. И. Языковая поддержка архитектурно-независимого параллельного программирования // Языки программирования и компиляторы : тр. конф. 2017. С. 169–172.

4. Технологические аспекты создания, преобразования и выполнения функционально-потоковых параллельных программ / А. И. Легалов [и др.] // Научный сервис в сети Интернет: все грани параллелизма : тр. Междунар. суперкомпьютерной конф. 2013. С. 443–447.

5. Особенности хранения функционально-потоковых параллельных программ / А. И. Легалов [и др.] // Вестник СибГАУ. 2013. № 4 (50). С. 53–57.

6. Событийная модель вычислений, поддерживающая выполнение функционально-потоковых параллельных программ / А. И. Легалов [и др.] // Системы. Методы. Технологии. 2012. № 1 (13). С. 113–119.

7. Романова Д. С. Организация тестирования для проверки корректности инструментальных средств языка Пифагор // Проспект Свободный – 2018 : материалы Междунар. конф. молодых учен. / Сиб. федер. ун-т. 2018. С. 25–28.

8. Удалова Ю. В. Верификация функционально-потоковых параллельных программ с помощью интервальных формул // Образовательные ресурсы и технологии. 2016. № 2 (14). С. 259–262.

9. Ушакова М. С., Легалов А. И. Верификация программ со взаимной рекурсией на языке Пифагор // Моделирование и анализ информационных систем. 2018. Т. 25, № 4 (76). С. 358–381.

10. Васильев В. С., Легалов А. И. Оптимизация инварианта цикла в языке Пифагор // Моделирование и анализ информационных систем. 2018. Т. 25, № 4 (76). С. 347–357.

11. A toolkit for the development of data-driven functional parallel programmes / А. И. Легалов [и др.] // Communications in Computer and Information Science. 2018. Т. 910. P. 16–30.

12. Инструментальная поддержка создания и трансформации функционально-потоковых параллельных программ / А. И. Легалов [и др.] // Тр. Ин-та систем. программирования РАН. 2017. Т. 29, № 5. С. 165–184.

13. Инструментальная поддержка формальной верификации программ, написанных на языке функционально-потокового параллельного программирования / М. С. Ушакова, А. И. Легалов // Вестник Южно-Уральского гос. ун-та. Серия: Вычислительная математика и информатика. 2015. Т. 4, № 2. С. 58–70.

14. Удалова Ю. В., Легалов А. И. Верификация функционально-потоковых параллельных программ методом индуктивных утверждений // Доклады АН ВШ РФ. 2014. № 2–3 (23-24). С. 125–132.

15. Люстерник Л. А. Математический анализ. Функции, пределы, ряды, цепные дроби. М. : ИЛ, 2012. 442 с.

Udalova Julia Vasilyevna – Ph. D., associate professor, Siberian Federal University. E-mail: judalova@sfu-kras.ru.

Kuzmin Dmitry Alexandrovich – Ph. D., associate professor, Siberian Federal University. E-mail: dkuzmin@sfu-kras.ru.

Удалова Юлия Васильевна – кандидат технических наук, доцент, Сибирский федеральный университет. E-mail: judalova@sfu-kras.ru.

Кузьмин Дмитрий Александрович – кандидат технических наук, доцент, Сибирский федеральный университет. E-mail: dkuzmin@sfu-kras.ru.