Научное программное обеспечение в образовании и науке

УДК 004.42 DOI 10.25559/SITITO.2017.4.446

Алексеев Е.Р.[1], Демин П.А.[2], Лутошкин Д.А.[1], Стародумов В.А.[1]

[1] Вятский государственный университет, г. Киров, Россия
[2] АО «НИИ СВТ», г. Киров, Россия

СВОБОДНЫЕ И ПРОПРИЕТАРНЫЕ КОМПИЛЯТОРЫ С(С++) И ФОРТРАНА ПРИ РАЗРАБОТКЕ ЭФФЕКТИВНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРИЛОЖЕНИЙ

Аннотация

Представленная авторами работа посвящена особенностям реализации различных алгоритмов вычислительной математики с помощью свободных и проприетарных компиляторов. Проведён анализ быстродействия последовательных и параллельных приложений на примере задач умножения матриц, решения систем линейных алгебраических уравнений точными и итерационными методами. Новые технологии параллельного программирования (автораспараллеливание, комассивы) реализованы исключительно в современных проприетарных компиляторах. Приведены результаты анализа быстродействия параллельных программ высокоточных вычислений, разработанных авторами. Выработаны рекомендации об особенностях реализации задач вычислительной математики с помощью различных языков программирования и компиляторов. Как свободные, так и проприетарные компиляторы позволяют разрабатывать высокоэффективные последовательные приложения. Преимущества проприетарных компиляторов проявляются при разработке параллельных приложений в первую очередь за счёт поддержки новейших технологий (автораспараллеливание и комассивы).

Ключевые слова

Вычислительная математика; компиляторы: g++, gfortran, ifort, icpc, pgc++, pgfortran; технологии параллельного программирования; комассивы; автораспараллеливание; MPI; OpenMP.

Alekseev E.R.[1], Demin P.A.[2], Lutoshkin D.A.[1], Starodumov V.V.[1]

[1] Vyatka State University, Kirov, Russia
[2] JSC «NII SVT», Kirov, Russia

THE FREE AND PROPRIETARY COMPILERS C(C++) AND FORTRAN AT DEVELOPMENT OF EFFECTIVE COMPUTING APPLICATIONS

Abstract

The work presented by authors is devoted to features of implementation of algorithms of calculus mathematics with the aid of free and proprietary compilers. The analysis of speed of consecutive and parallel applications was carried out on the example of task of matrix multiplication, the solution of systems of the linear algebraic equations by exact and iterative methods. New technologies of parallel programming (auto parallelization, coarrays) are implemented only in modern proprietary compilers. Results of the analysis of high-speed performance of parallel programs of the high-precision calculations developed by authors are presented. Recommendations about the features of realization of task of calculus mathematics were worked out with the aid of various programming languages and compilers. Free and proprietary compilers allow to develop highly effective consecutive applications. Advantages of proprietary compilers are shown when developing parallel applications first of all due to support of the new technologies (auto parallelization, coarrays).

Keywords

Calculus mathematics; compilers: g++, gfortran, ifort, icpc, pgc++, pgfortran; technologies of parallel programming; coarrays; auto parallelization; MPI; OpenMP.

Введение

Разработка приложений, предназначенных для решения задач вычислительной математики имеет ряд особенностей:

- в этих приложениях производится большое количество вычислений;
- для работы программ необходимо большое количество памяти;
- программы содержат графический вывод результатов.

Традиционно для решения задач вычислительной математики используются два класса программ:

- классические компиляторы;
- математические программы.

К преимущества математическим программ можно отнести быстроту разработку приложений и большое количество готовых программных модулей для решения стандартных подзадач. К недостаткам – достаточно долгое время работы полученных приложений, сложная переносимость, большой объём (зачастую весь программный продукт разработчика необходим для запуска отдельного приложения).

При разработке вычислительных задач с помощью классических компиляторов ситуация прямо противоположная. Разработчик тратит большое количество времени на разработку, но получает достаточно компактное и быстроработающее приложение.

Выбор, какой класс программ использовать зависит от конкретных условий и задачи, которую необходимо решить.

Современные проприетарные и свободные компиляторы

В данной работе представлен сравнительный анализ возможностей и быстродействия свободных и проприетарных компиляторов C(C++) и Фортрана, используемых для решения инженерных и вычислительных задач. Фортран и C(C++) являются основными языками, используемыми математиками и инженерами для решения прикладных задач.

Одним из важных вопросов в сложных вычислительных задачах является вопрос эффективности и быстродействия.

В данной работе будет проведёт анализ следующих компиляторов:

- свободные компиляторы gcc, g++, gfortran [1], которые присутствуют в репозиториях большинства современных дистрибутивов Linux; эти компиляторы поддерживают большинство современных стандартов языка и формируют высокоэффективный код; компиляторы портированы в Windows (проект mingw);
- высокоэффективные компиляторы C(C++) и Фортрана компании Intel, входящие в состав комплекса проприетарных программ Intel Parallel Studio; в некоммерческих целях компиляторами могут пользоваться бесплатно студенты, преподавали и научные работники [2];
- проприетарные компиляторы компании Pgroup; существует бесплатная версия компиляторов семейства PGI – PGI Community Edition [3].

При тестировании использовались компиляторы Фортрана – gfortran-7 (7.2), pgfortran (17.4), ifort (Intel Parallel Studio 2017 update4) и C(C++) (компиляторы g++ (7.2), pgc++ (17.4), icpc (Intel Parallel Studio 2017 update4). Тестовый компьютер для последовательного кода: узел кластера ВятГУ: 8 ядер – Intel (R) Xeon(R) CPU E5345 @ 2.33GHz, ОЗУ – 4 Гб, ОС Ubuntu 14.04.5 LTS.

Многие инженерные и математические задачи сводятся к одной или несколькими задачам линейной алгебры. Кроме того, одной из наиболее часто встречающихся задач является задача численного интегрирования. Поэтому авторами было принято решение в качестве тестовых использовать следующие задачи:

- умножение матриц (классический и блочный алгоритмы);
- решение системы линейных алгебраических уравнений (метод Гаусса, Гаусса-Жордана; простой итерации; метод Зейделя);
- численного интегрирования.

Анализ быстродействия последовательных вычислительных приложений

Одной из классических проблем вычислительной математики является задач умножения матриц. Вариантов записи кода классического алгоритма умножения матриц – 6, можно менять порядок циклов (ijk) местами [4]. Результаты тестирования приведены ниже.

Как видно из табл. 1-2, при использовании проприетарных компиляторов, не важно в каком порядке записывать циклы в коде программ.

Компиляторы ifort, icpc, pgfortran, pgc++ достаточно хорошо оптимизируют исходный код и формируют оптимальное приложение.

Таблица 1. Быстродействие программ (время, с), реализующих классический алгоритм на Фортране с использованием свободных и проприетарных компиляторов при различном порядке циклов

		Порядок выполнения циклов						
	Время, t(c)	ijk	jik	kij	ikj	jki	kji	matmul
1024	gfortran	43.33	51.84	122.73	114.43	2.48	3.7	0.62
	ifort	2.64	2.61	2.59	2.65	2.63	2.67	2.67
	pgfortran	2.27	2.26	2.27	2.27	2.28	2.27	0.69
1536	gfortran	163.67	169.36	357.55	365.71	8.5	13.91	1.88
	ifort	9.11	9.22	9.13	9.2	9.12	9.14	9.22
	pgfortran	8.02	8.07	7.97	8.1	8.13	7.97	2.69

Таблица 2. Быстродействие программ, реализующих классический алгоритм на C(C++) с использованием свободных и проприетарных компиляторов

		Порядок выполнения циклов					
	Время, t(c)	ijk	jik	kij	ikj	jki	kji
1024	g++	12.18	10.53	4	2.79	20.29	18.12
	icpc	2.92	2.97	2.97	2.97	2.97	2.97
	pgc++	13.32	11.88	4.36	4.06	21.1	20.51
1536	g++	46.07	44.92	14.44	9.42	53.51	55.82
	icpc	10.02	10.01	10.01	10.05	10.02	10.01
	pgc++	55.75	37.39	15.21	13.55	55.53	61.74

Таблица 3. Быстродействие программ, реализующих блочный алгоритм на C(C++) и Фортране (размер блока = 16) с использованием свободных и проприетарных компиляторов

Размерность системы	Компилятор	Время счёта, с	Компилятор	Время счёта, с
1024	gfortran	0.81	g++	2.65
	ifort	1.95	icpc	1.65
	pgfortran	0.92	pgc++	2.76
1536	gfortran	2.46	g++	8.88
	ifort	5.79	icpc	5.57
	pgfortran	2.67	pgc++	9.56

При использовании свободных компиляторов g++, gfortran программист должен аккуратно писать код, записывая циклы в оптимальном для данного языка порядке. Однако, при оптимальном написании текста программ задач, подобных умножению матриц, формируемое с помощью свободного компилятора приложение не уступает по быстродействию программным продуктам, генерируемым с помощью проприетарных компиляторов ifort, icpc, pgfortran, pgc++. А иногда (см. табл. 1-2) программы, сгенерированные свободными компиляторами, превосходят по быстродействию свои проприетарные аналоги. Обращает на себя внимание оптимизация встроенных подпрограмм, проведенная в свободном компиляторе Фортрана седьмой версии (gfortran-7). Например, код подпрограммы matmul, сгенерированный с помощью gfortran, работает значительно быстрее кода, получаемого с помощью ifort (см. табл. 1).

Авторами был проведён анализ быстродействия блочного алгоритма умножения матриц. Результаты теста представлены в табл. 3. При реализации блочного алгоритма на Фортране использовалась встроенная оптимизированная

процедура matmul, что привело к значительному уменьшению времени работы свободного приложения блочного умножения на Фортране. Фортран-программа, реализующая блочный алгоритм умножения, которая была сгенерирована с помощью gfortran-7, работает быстрее не только своих проприетарных аналогов, но и значительно быстрее любых программ, написанных на C(C++). Эта же тенденция сохраняется и в алгоритме Штрассена [5].

Ещё одной часто встречающей в практике задачей является решение системы линейных алгебраических уравнений (СЛАУ) итерационными методами. В качестве примера рассмотрим решение СЛАУ большой размерности методом простой итерации и методом Зейделя.

При разработке вычислительных программ обработки матриц большой размерности следует учитывать, что матрицы на Фортране в памяти хранятся по столбцам, на C(C++) – по строкам.

При анализе быстродействия СЛАУ была выбрана следующая тестовая система линейных алгебраических уравнений $A \cdot x = b$, где $A_{i,j} = \begin{cases} 2n, i = j \\ 1, \ i \neq j \end{cases}$, $b_i = \frac{n(n+1)}{2} + i(2n-1)$ [6]. Ключевыми особенностями этой системы являются: диагональное преобладание матрицы A, что позволяет её использовать в качестве тестовой для итерационных методов (Зейделя, Якоби и ряда других); заранее известно решение системы любой размерности $x = \begin{pmatrix} 1 \\ 2 \\ \dots \\ n \end{pmatrix}$. Был проведен анализ быстродействия программ на Фортране и C(C++). Код на Фортране был написан в двух вариантах: с использованием матричных операций (1) и с использованием циклов (2). В таблицах 4-5 представлено время счёта на Фортране и C(C++) соответсвенно.

Как видно из результатов, представленных в табл. 4-5, создаваемые с помощью компиляторов gfortran и pgfortran, программы на Фортране (компи) при использовании матричных подпрограмм и операций работают значительно быстрее. Однако оптимизация циклов, используемая компилятором Фортрана компании Intel, позволяет приложению с циклами работать быстрее на 15-20%.

В таблице 6 представлено время счёта программы решения СЛАУ методом Гаусса на языках C(C++) и Фортране с помощью свободных и проприетарных компиляторов.

Таблица 4. Время счёта программ на Фортране решения СЛАУ итерационными методами[26]

Размерность системы	Компилятор	Метод Зейделя			Метод простой итерации		
		k	1	2	k	1	2
12000	gfortran	13	13.6	11.4	26	36.13	13.62
	pgfortran		13.37	11.16		31.38	13.09
	ifort		9.76	11.21		24.38	13.73
15000	gfortran	13	17.49	14.15	26	56.07	20.56
	pgfortran		17.22	13.51		37.88	19.61
	ifort		12.21	13.85		29.12	22.17

Таблица 5. Время счёта программ на C(C++) решения СЛАУ итерационными методами[27]

Размерность системы	Компилятор	Метод Зейделя		Метод простой итерации	
		k		k	
12000	g++	12	4.25	34	12.04
	pgc++		4.04		11.46
	icpc		4.19		11.89
15000	g++	12	6.64	34	18.8
	pgc++		6.32		17.9
	icpc		6.55		18.56

[26] Здесь k – количество итераций, 1 – программа без использования матричных операций Фортрана, 2 – программа с использованием матричных операций, ключ компиляции -O2, точность 0.000001.

[27] Здесь k – количество итераций, ключ компиляции -O2, точность 0.000001.

Таблица 6. Время счёта программ на Фортране и C(C++) решения СЛАУ методом Гаусса

Размерность системы	Компилятор Фортрана	Время счёта, с	Компилятор C(C++)	Время счёта, с
1000	gfortran	3.78	g++	0.89
	ifort	3.9	icpc	0.88
	pgfortran	1.44	pgc++	0.85
2000	gfortran	34.37	g++	10.10
	ifort	33.02	icpc	10.11
	pgfortran	16.13	pgc++	9.95

Таблица 7. Время счёта программ на Фортране и C(C++) решения задачи численного интегрирования

	Компилятор Фортрана	Время счёта, с	Компилятор C(C++)	Время счёта, с
Метод прямоугольников	gfortran	0.8	g++	0.8
	ifort	0.4	icpc	0.4
	pgfortran	0.8	pgc++	0.85
Метод трапеций	gfortran	1.6	g++	1.55
	ifort	0.8	icpc	0.83
	pgfortran	1.7	pgc++	1.45
Метод Симпсона	gfortran	3.1	g++	3.2
	ifort	1.6	icpc	1.5
	pgfortran	3.3	pgc++	3.3

В задачах, подобных решению системы методом Гаусса (решение СЛАУ методом Жордана-Гаусса, вычисление обратной матрицы методом Жордана Гаусса и др.) код на C(C++) работает значительно быстрее, чем на Фортране. На это оказывает влияние оптимизация циклов, используемая в всех компиляторах C(C++). Все исследуемые компиляторы C(C++) формируют приложения, работающие примерно с одинаковой скоростью. Код на Фортране работает значительно медленнее. В алгоритме негде применить основное преимущество современных компиляторов Фортрана – матричные и конвейерные операции. Обращает на себя внимание оптимизация встроенных циклов, проведенная разработчиками компилятора Фортрана компании Pgroup. Это позволило ускорить код в подобных приложениях в 2-3 раза по сравнению с кодом, генерируемым gfortran и ifort. В таблице 7 представлено время счёта программы численного интегрирования на языках Фортране и C(C++).

Кроме последовательных и конвейерных приложений решения задач вычислительной математики интерес представляет сравнительный анализ свободных и проприетарных компиляторов при построении параллельных приложений.

Разработка параллельных приложений с помощью технологии автораспараллеливания

Одним из новейших методов построения параллельных программ является технология автораспараллеливания, полноценная поддержка которой реализована в только проприетарных компиляторах Intel посредством опций командной строки **-parallel**. Компилятор анализирует код программы и определяет, какие циклы имеет смысл выполнять параллельно. В цикле, который подлежит автораспараллеливанию, должны соблюдаться следующие условия:

- до начала выполнения программы количество итераций должно быть определено;
- не должно быть аварийных выходов из цикла, и входа внутрь цикла с помощью оператора goto;
- итерации цикла должны быть независимыми; при использовании

итерационных алгоритмов, подобных суммированию, компилятор ifort самостоятельно преобразовывает цикл, исключая зависимости между итерациями.

В таблице 8 приведены результаты применения опции автораспараллеливания к последовательным программам [28] умножения матриц (классический алгоритм) на языке Фортран и C(C++).

Таблица 8. Автораспараллеливание в программах умножения матриц на Фортране

	Фортран			C(C++)	
	Размерность матрицы			Размерность матрицы	
	2000	4000		2000	4000
-ifort	4.4	37	-icpc	5.21	41.73
ifort -parallel	2	24	icpc -parallel	2.32	26.35

Таблица 9 – Быстродействие параллельных высокоточных приложений

Количество ядер	intel	gcc	pgi
1	56,82	54,74	60,04
2	28,58	27,56	31,40
4	16,69	16,77	16,14
8	8,88	8,56	9,45
16	8,34	9,07	8,52

Как видно из приведенных в табл. 8 данных, технология автораспараллеливания находится только в начале своего развития, её поддержка начинает осуществляться в компиляторах компании Intel. Однако авторы предполагают, что автораспараллеливание будет и дальше развиваться, и, вероятно, поддержка этого направления может появиться и в свободных компиляторах.

Разработка параллельных приложений с помощью MPI и OpenMP

Современные компиляторы C(C++) и Фортрана поддерживают классические технологии параллельного программирования MPI и OpenMP [6-9].

Приведём результаты тестирования разработанной авторами параллельной программы на языке C++ высокой точности (библиотека MPFRC++) решения СЛАУ методом простой итерации. В таблице 9 приведены результаты быстродействия (точность 10^{-20}, размер тестовой матрицы 1500x1500, длина мантиссы 72 знака).

Использование восьми ядер обеспечило ускорение в 6.5 раз на всех компиляторах. При дальнейшем увеличении числа ядер эффективность ускорения начинает падать. Как видно из результатов, приведенных в таблице 9, существенной разницы по быстродействию между параллельными MPI-приложениями, полученными с помощью разных компиляторов нет. Эта же тенденция проявлялась и в других приложениях. Принципиальным отличием были более быстрые базовые алгоритмы, полученные с использованием компилятора gfortran-7 (за счёт оптимизации матричных операций подобных matmul), но ускорение при распараллеливании оставалось таким же.

При использовании параллельных программ, разработанных с использованием технологии OpenMP, значительных отличии при использовании компиляторов gfortran, pgfortran, ifort, g++, pgc++, icpc выявлено не было.

Разработка параллельных приложений на Фортране с помощью комассивов

Кроме поддерживаемых в других языках программирования технологий параллельного программирования OpenMP и MPI, в новый стандарт языка Fortran-2008 включены встроенные средства распараллеливания Co-Arrays (комассивы), которые могут быть реализованы в системах как с распределенной, так и с общей памятью. Программа, содержащая комассивы, выполняется асинхронно несколько раз. Каждая работающая копия программы (image) имеет свои локальный набор данных. Для получения максимальной

28 Программы запускались на локально ПК с характеристиками Intel(R) Core(TM) i5-2500 CPU 4x3.30GHz, ОЗУ – 20 ГГб, ОС Linux Mint 18.2 x64, компиляторы gfortran-7 (7.2), 6.2), pgfortran (17.4), ifort (Intel Parallel Studio 2017 update2).

производительности число работающих копии программы не должно превышать количество процессоров. Полноценная поддержка комассивов реализована только в компиляторе Фортрана компании Intel, более подробно об этом в работах [5, 11, 12].

Вывод графической информации в консольных приложениях на С(С++) и Фортране

При использовании консольных компиляторов С(С++) и следует учитывать, что отсутствуют средства графического вывода информации. Был проведён анализ возможностей сторонних графических библиотек, в результате которого представлены [13]. Было выявлено, что основными проблемами при использовании при использовании сторонних графических библиотек являются: плохая переносимость в различные операционные системы, большой размер, сложность подключения графических библиотек.

Авторами были разработаны универсальные кроссплатфоменные библиотеки на языках С++ и Фортран представления результатов вычислений в графическом виде с использованием графических библиотек OpenGL и FreeGLUT.

Разработанные библиотеки содержат программные средства, позволяющие строить строить точечные графики, графики двух- и трёх-мерных функций, плоскости в трёхмерном пространстве. После получения изображения, пользователь может управлять положением камеры: смещать камеру по осям, вращать её, масштабировать изображение. Подробно работа представлена авторами на 7-й конференции-конкурсе «Инновационные информационно-педагогические технологии в ИТ-образовании» 24-26 ноября 2017 года на ВМК МГУ.

Заключение

При разработке последовательных вычислительных приложений можно использовать как свободные, так и проприетарные компиляторы Фортрана и С(С++). При использовании свободных компилятора g++ в задачах обработки матриц большой размерности следует учитывать, что матрицы в памяти хранятся построчно. Современные проприетарные компиляторы языка С++ (icpc, pgc++) не дают выигрыша по времени по сравнению со свободным компилятором g++. Однако, следует помнить, что в состав Intel Parallel Studio, кроме компиляторов входит и проприетарная библиотека MKL – это проприетарная библиотека многопоточных математических функций на языках С/С++, Fortran.

При написании программ обработки матриц большой размерности на Фортране следует учитывать, что матрицы в Фортране хранятся по столбцам; в Фортране присутствует большое количество матричных операций, которые формируют код, который может обрабатываться процессором конвейерно; матричные подпрограммы в свободных компиляторах значительно оптимизированы. Учёт способа хранения данных в Фортране, использование матричных подпрограмм и операций программах позволяет построить с помощью свободного компилятора gfortran приложения, не уступающие по быстродействию программам, сгенерированными проприетарными трансляторами. А в некоторых случаях, например, при написании программ умножения матриц, используя блочный алгоритм и алгоритм Штрассена свободные компиляторы формируют и более быстроработающие приложения.

При разработке параллельных приложений проприетарные компиляторы предпочтительнее, чем свободные. Это связано с возможностью использование технологии автораспараллеливания при генерации параллельного кода, а также реализацией технологии комассивов в компиляторе Фортрана компании Intel.

Что касается классических технологий параллельного программирования OpenMP и MPI, то они поддерживаются всеми рассматриваемыми в статье компиляторами.

Перед разработчиком инженерного или математического приложения, связанного с большим количеством вычислений, стоит проблема выбора компилятора. На основании проведенного авторами анализа можно предложить следующие рекомендации.

1. При использовании свободных компиляторов С(С++) и Фортрана в коде необходимо учитывать особенности хранения матриц в памяти компьютера. В С(С++) – матрицы хранятся построчно, в Фортране – по столбцам. Учёт этого оказывает существенное влияние на быстродействие программ обработки матриц (см. табл. 1-2). Проприетарные компиляторы самостоятельно оптимизируют расположение в памяти. При использовании языка программирования Фортран надо максимально задействовать встроенные матричные подпрограммы и операции.

Особенную эффективность это может дать с учётом оптимизации матричных подпрограмм, реализованной в последней версии компилятора gfortran-7 (см. табл. 3). Если принимать во внимание выше перечисленные особенности при программировании задач вычислительной математики, то свободные компиляторы в последовательных приложениях мало в чём уступают своим проприетарным аналогам.

2. При разработке параллельных приложений, к преимуществам проприетарных компиляторов (в первую очередь компиляторов Intel) можно отнести поддержку новейших технологий таких, как автораспараллеливание и комассивы. При использовании классических технологий параллельного программирования таких, как MPI и OpenMP, возможности проприетарных и свободных компиляторов примерно одинаковы.

Литература

1. GCC, the GNU Compiler Collection- GNU Project - Free Software Foundation (FSF). [Электронный ресурс] URL: https://gcc.gnu.org (Дата обращения 13.07.2017).
2. Получите право на бесплатное программное обеспечение | Intel® Software [Электронный ресурс] URL: https://software.intel.com/ru-ru/qualify-for-free-software/educator (Дата обращения 13.07.2017).
3. PGI | Products | Community Edition. [Электронный ресурс] URL: http://www.pgroup.com/products/community.htm (Дата обращения 13.07.2017).
4. П.А. Демин, Е.Р. Алексеев. Матричные операции языка Fortran // ОБЩЕСТВО. НАУКА. ИННОВАЦИИ (НПК-2017) [Электронный ресурс] : сб. статей : Всерос. ежегод. науч.-практ. конф., 1–29 апреля 2017 г. – Киров : [Науч. изд-во ВятГУ], 2017. – С.979-990.
5. Алексеев Е.Р., Соболева О.В. Современный язык программирования Фортран в образовании и научных исследованиях // Современные информационные технолгии и ИТ-образование, 2016, т. 12, №4. С.110-116.
6. Лупин С.А., Посыпкин М.А. Технологии параллельного программирования. – М: ИД «Форум»: ИНФРА-М, 2014.
7. Богачёв К.Ю. Основы параллельных вычислений. – М: БИНОМ. Лаборатория знаний, 2013.
8. Антонов А.С. Технологии параллельного программирования MPI и OpenMP. – М: Издательство Московского университета, 2012.
9. Старченко А.В., Данилкин Е.А., Лаева В.И., Проханов С.А. Практикум по методам параллельных вычислений. – М: Издательство Московского университета, 2010.
10. Левин М.П. Параллельное программирование и использование OpenMP. – М.: ИНТУИТ; БИНОМ. Лаборатория знаний, 2008.
11. Горелик А.М. Программирование на современном Фортране. – М.: Финансы и статистика, 2006.
12. Арьен Маркус. Современный Фортран на практике. – М: ДМК-Пресс, 2015.
13. Алексеев Е. Р., Демин П. А., Костюк Д. А. Возможности графического вывода результатов в последовательных и параллельных кроссплатформенных вычислительных приложениях на Фортране и C(C++) // Advanced science, 2017, №3.

References

1. GCC, the GNU Compiler Collection- GNU Project - Free Software Foundation (FSF). [Электронный ресурс] URL: https://gcc.gnu.org (Дата обращения 13.07.2017).
2. Poluchite pravo na besplatnoe programmnoe obespechenie | Intel® Software [Электронный ресурс] URL: https://software.intel.com/ru-ru/qualify-for-free-software/educator (Дата обращения 13.07.2017).
3. PGI | Products | Community Edition. [Электронный ресурс] URL: http://www.pgroup.com/products/community.htm (Дата обращения 13.07.2017).
4. P.A. Demin, E.R. Alekseev. Matrichnie operacii yazika Fortran // OBSCHESTVO. NAUKA. INNOVACII (NPK-2017) [Электронный ресурс]: sb. statei: Vseros. ejegod. nauch. - prakt. konf., 1–29 aprelya 2017 g. – Kirov: [Nauch. Izd-vo VyatGU], 2017. – C.979-990.
5. Alekseev E. R., Soboleva O. V. Sovremenniy yazik programmirovaniya Fortran v obrazovanii i nauchnih issledovaniyah // Sovremenie informatsionnie tehnologii i IT-obrazovanie, 2016, t. 12, №4. C.110-116.
6. Lupin S.A., Posipkin M.A. Tehnologii parallelnogo programmirovaniya. – M: ID «Forum»: INFRA-M, 2014.
7. Bogachev K.Yu. Osnovi parallelnih vichislenii. – M: BINOM. Laboratoriya znanii, 2013.
8. Antonov A.S. Tehnologii parallelnogo programmirovaniya MPI i OpenMP. – M: Izdatelstvo Moskovskogo universiteta, 2012.
9. Starchenko A.V., Danilkin E.A., Laeva V.I., Prohanov S.A. Praktikum po metodam parallelnih vichislenii. – M: Izdatelstvo Moskovskogo universiteta, 2010.
10. Levin M.P. Parallelnoe programmirovanie ispolzovanie OpenMP. – M.: INTUIT; BINOM. Laboratoriya znanii, 2008.
11. Gorelik A.M. Programmirovanie na sovremennom Fortrane. – M.: Finansi i statistika, 2006.
12. Aren Markus. Sovremennii Fortran na praktike. – M: DMK-Press, 2015.
13. Alekseev E.R., Demin P.A., Kostyuk D.A. Vozmozhnosti graficheskogo vivoda rezul'tatov v posledovatel'nih i parallel'nih krossplatfomennih vicheslitel'nih prilozheniyah na Fortrane i C(C++) // Advanced science, 2017, №3.

Поступила: 30.09.2017

Об авторах:

Алексеев Евгений Ростиславович, кандидат технических наук, доцент, профессор кафедры фундаментальной информатики и прикладной математики, Вятский государственный университет, er.alekseev@yandex.ru

Демин Петр Александрович, инженер-программист, АО НИИ СВТ, demin-rabota@yandex.ru

Лутошкин Денис Александрович, магистрант, Вятский государственный университет, disa1995@mail.ru
Стародумов Вячеслав Владимирович, магистрант, Вятский государственный университет, WStarD@citydom.ru

Note on the authors:
Alekseev Evgeniy R., PhD in Technical Sciences, associate professor, professor of Department of Fundamental Informatics and Applied Mathematics, Vyatka State University, er.alekseev@yandex.ru
Demin Petr A., software engineer, JSC «NII SVT», demin-rabota@yandex.ru
Lutoshkin Denus A., master's degree student, Vyatka State University, disa1995@mail.ru
Starodumov Vyacheslav V., master's degree student, Vyatka State University, WStarD@citydom.ru