УДК 004.432

Алексеев Е.Р.[1], Соболева О.В.[2]

[1] Вятский государственный университет, г. Киров, Россия
[2] ООО «Студия «Атум», г. Киров, Россия

СОВРЕМЕННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ ФОРТРАН В ОБРАЗОВАНИИ И НАУЧНЫХ ИССЛЕДОВАНИЯХ

АННОТАЦИЯ

В статье проведён анализ современных стандартов и реализаций Фортрана. Рассмотрены возможности языка при решении вычислительных задач. Обсуждается необходимость преподавания Фортрана в университетах.

КЛЮЧЕВЫЕ СЛОВА

Фортран, стандарт, компиляторы, gfortran, ifort, конвейерные операции, численные методы, параллельное программирование, комассивы.

Alekseev E.R.[1], Soboleva O.V.[2]

[1] Vyatka State University, Kirov, Russia
[2] Studio Atum, Kirov, Russia

THE MODERN LANGUAGE OF PROGRAMMING FORTRAN IN EDUCATION AND SCIENTIFIC RESEARCH

ABSTRACT

In the article the modern standards and implementations of the FORTRAN are analyzed. The possibilities of language in case of solving of computing tasks are considered. Necessity of teaching the FORTRAN for universities is discussed.

KEYWORDS

Fortran; standard; compilers; gfortran; ifort; conveyor operation; numerical methods, parallel programming, co-arrays.

Фортран – один из самых первых языков программирования высокого уровня. Он разрабатывался в первую очередь для решения вычислительных задач. Современный язык также ориентирован на решение задач, связанных с большим количеством вычислений. За более чем пятидесятилетнюю историю он значительно изменился. Подробнее о развитии Фортрана описано в [1,2].

Язык сохранил основные черты: простота кода, хорошая читаемость программ; высокая скорость компиляции и выполнения программ; быстрота первоначального освоения; совместимость с предыдущим синтаксисом.

Главное преимущество современных версий языка – поддержка новых технологий программирования, таких как объектно-ориентированное и параллельное программирование; высокая эффективность получаемых приложений.

Среди основных особенностей синтаксиса современного Фортрана можно выделить следующие [3,4]:

- поддержка шестнадцатибайтных целых, вещественных чисел, тридцатидвухбайтных комплексных чисел;
- массивы переменной длины;
- встроенные операции с массивами;
- поддержка и реализация конвейерных операций;
- реализация параллельных вычислений; кроме классического фортрановского цикла do в последних версиях стандарта (2003, 2008) появился цикл forall, который выполняется параллельно.

Благодаря наличию операций с массивами, конвейерных операций и средств параллельного

программирования компиляторы Фортрана генерируют высокоэффективный исполняемый код[2].

Можно выделить следующие компиляторы языка Фортран:

1. В состав репозиториев большинства дистрибутивов Линукс входит семейство компиляторов gcc. Gcc поддерживает и современные стандарты Фортран. Свободный компилятор Фортрана gfortran требует наличия компилятора C(c++) на ПК. Исполняемый код получается посредством промежуточного кода на C.
2. Компания Intel разрабатывает высокоэффективный автораспараллеливающийся кроссплатформенный компилятор Фортран, который входит в состав комплекса программ Intel Parallel Studio. Компиляторы Intel доступны студентам и преподавателям университетов бесплатно.
3. Бесплатный свободный кроссплатформенный компилятор g95 (http://www.g95.org/).
4. Компилятор Фортран компании Oracle, входящий в состав Oracle Developer Studio (http://www.oracle.com/technetwork/server-storage/developerstudio/overview/index.html). Компилятор ориентирован в первую очередь на использование в Oracle Linux (http://www.oracle.com/ru/technologies/linux/overview/index.html).

В данной работе приводятся результаты, полученные авторами с использованием свободного компилятора gfortran (версии 5.4, 6.2), компилятора ifort (Intel Parallel studio 17). Эти компиляторы выбраны исходя из их доступности, универсальности, возможности формирования высокоэффективного кода для различных операционных систем. Именно эти компиляторы используются на многих вычислительных кластерах.

Возможность использования Фортрана при решении исследовательских вычислительных задач и при обучении студентов инженерного, математического и компьютерного направления обусловлена следующими основными факторами [5,6]:

1. быстрое освоение синтаксиса языка;
2. хорошо читаемый простой код;
3. из исходного кода может быть получено высокоэффективное кроссплатформенное приложение.

Ряд инженерных и экономических задач сводится к задачам линейной алгебры большой размерности (умножение матриц, решение систем линейных алгебраических уравнений, вычисление обратной матрицы, вычисление определителя и др.).

Рассмотрим насколько эффективно использование Фортрана при решении подобных задач.

В Фортране есть ряд встроенных функций [4-6] для операции над массивами и матрицами.

1. Функция умножения матриц matmul(a,b) вычисляет $C = A \cdot B$.

2. Функции maxval(a[, mask=логическое условие]) и minval(a[, mask=логическое условие]) находят максимальное и минимальное значение в матрице (массиве) при выполнении некоторого условия. Пример использования

! Найти минимальное чётное число в массиве среди нечётных элементов массива.

minval(x(1:25:2), mask=x mod 2 == 0)

3. Функции maxloc(a[, mask=логическое условие]), minloc(a[, mask=логическое условие]) вычисляют индекс максимального и минимального значений в массиве при выполнении некоторого условия.

4. Функция sum(a[, mask=логическое условие]) вычисляет сумму элементов массива при выполнении некоторого условия.

5. Функции lbound(x) и ubound(x) возвращают одномерные массивы нижних и верхних значений индексов всех измерений массива x.

6. Функция transponse(a) возвращает транспонированную матрицу.

Умножение матриц. Рассмотрим использование классического и блочного алгоритма умножения матриц при программировании на Фортране.

Классический алгоритм формирует каждый элемент матрицы $C = A \cdot B$ по формуле $C_{i,j} = \sum_{k=1}^{n} A_{i,k} \cdot B_{k,j}$. При программировании подобного выражения на любом языке программирования не следует забывать об эффективном способе написания тройного цикла. С учётом того, что матрицы в Фортране хранятся по столбцам, внутренний цикл должен быть по переменной i, а на языке C имеет смысл его организовать по переменной j.

В блочном алгоритме умножения матриц каждая матрица может быть разделена на k^2 блоков

2 При решении ряда вычислительных задач код генерируемый компиляторами Фортрана является самым быстрым.

$M = \begin{pmatrix} M_{1,1} & M_{1,2} & \dots & M_{1,k} \\ M_{2,1} & M_{2,2} & \dots & M_{2,k} \\ \dots & \dots & \dots & \dots \\ M_{k,1} & M_{k,2} & \dots & M_{k,k} \end{pmatrix}$, где $M_{i,j}$ – матрица размерност и $N \times N$ k=n/N, где N – целое число. Каждый блок $C_{i,m}$ матрицы C может быть вычислен, как сумма произведений соответствующих блоков матриц A и B.

$$C_{i,m} = \sum_{j=1}^{k} A_{i,j} B_{j,m},\ i = 1,...,k,\ m = 1,...,k .$$

Результаты тестирования программ умножения матриц приведены в таблице[3] 1, а программ умножения матриц с использованием блочного алгоритма – в таблице 2.

Таблица 1. Время выполнения (в с.) программ умножения матриц, использующих классический алгоритм

N	Классический алгоритм, C		Классический алгоритм, Фортран		Matmul, Фортран	
	g++ (5.4)	icpc (17)	gfortran(5.4)	ifort (17)	gfortran(5.4)	ifort (17)
1024	1	0.6	0.82	0.58	0.63	0.6
2048	8.8	5	6.69	5.02	5.24	5.16
4096	76	40.46	56.58	41.84	44.59	43.48

Таблица 2. Время выполнения (в с.) программ умножения матриц, использующих блочный алгоритм

N	Блочный алгоритм, C		Блочный алгоритм, Фортран	
	g++ (5.4)	icpc (17)	gfortran(5.4)	ifort (17)
1024	1.66	0.85	0.42	0.36
2048	7.1	5	3.5	2.8
4096	57.9	35	30	22.8

Из полученных результатов можно сделать следующие выводы:

- код на Фортране хорошо читаем, его проще писать и значительно проще отлаживать, чем код на C(C++);
- программы умножения матриц, сгенерированные компиляторами с языка Фортран, работают быстрее, чем программы, для генерации которых использовались компиляторы C;
- современные компиляторы компании Intel генерируют более быстрый код, чем свободные компиляторы семейства gcc;
- компиляторы Intel оптимизируют организацию тройного цикла в матричном умножении (как и во многих других подобных случаях) и программист может не заниматься этим;
- при использовании свободных компиляторов неэффективная запись тройного цикла может привести к замедлению времени выполнения программы в 2-20 раз;
- исполняемый файл программ умножения, который получен с помощью компиляторов Intel может работать быстрее, чем код, полученный с помощью функции matmul.

Ещё более эффективно использование Фортрана при программировании итерационных алгоритмов решения систем линейных алгебраических уравнений. Особый интерес с точки зрения эффективности кода на Фортране является сравнение метода Зейделя и метода простой итерации.

Основная часть кода метода Зейделя

```
do while(r>eps)
k=k+1
x1=x
do i=1,n
```

3 Тестовый ПК, процессор Intel I5, 4x3.3 Ггц, ОЗУ – 4Ггб, ОС Linux Mint 18, ядро версии 4.4, компиляторы g++ 5.4, gfortran 5.4, icpc 17, ifort 17, ключ оптимизации -O2.

```
x(i)=bet(i)+dot_product(alf(i,:),x)
end do
r=maxval(abs(x1-x))
end do
```

Основная часть кода метода простой итерации

```
do while(r>eps)
k=k+1
x1=bet+matmul(alf,x)
r=maxval(abs(x1-x))
x=x1
end do
```

Как видно из приведенных фрагментов в методе простой итерации используются только матричные (т. е. конвейерные) операции, а в методе Зейделя присутствует цикл do, который никаким образом не может быть заменён конвейерной операцией. Поэтому гарантировано метод Зейделя будет считать медленнее. Из-за этого и невозможно качественно распараллелить метод Зейделя. Результаты тестирования подтверждают сделанные выводы.

Таблица 3. Время выполнения (в с.) программ решения СЛАУ итерационными методами

N	Метод простой итерации		Метод Зейделя	
	gfortran	ifort	gfortran	ifort
5000	0.32	0.24	0.56	0.55
10000	1.45	1.35	2.5	2.4

Таким образом напрашивается вывод. **Если существует возможность написать код на Фортране только с использованием матричных (конвейерных операций), то данный алгоритм может быть эффективно распараллелен и с использованием других технологий параллельного программирования (например, с помощью технологии MPI)**.

В компиляторах gfotran и ifort поддерживаются такие технологии параллельного программирования, как OpenMP (системы с общей памятью) и MPI (системы с распределённой памятью). Кроме того, в новый стандарт языка Fortran 2008 включены встроенные средства распараллеливания Co-Arrays (комассивы) [3], 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 свой локальный набор данных. Для получения максимальной производительности число работающих копий программы не должно превышать количество процессоров.

Для объявления комассива к обычному описанию добавляется служебное слово codimension и квадратные скобки. Оператор

```
real(8), codimension [*] :: x(1000), y(1000)
```

объявляет два вещественных комассива x и y размером 1000, которые определены в каждой копии (image) программы.

К данным, объявленным как комассив, можно обратиться из любой копии программы. В нашем примере обращение y(13) [3] означает значение y(13) в третьей копии исполняемой программы. Обращение к данным, описанным как комассив, без квадратных скобок относится к текущей копии программы. Например, x(100) – сотый элемент массива x в текущей копии. Передача данных между копиями программы (аналог пересылки сообщений в MPI) осуществляется с помощью обычного оператора присваивания.

В новом стандарте Фортрана присутствуют функции [3] num_images() – определение числа копий и this_image() – номер текущей копии[4] программы.

В Fortran 2008 введены [3]:

- операторы синхронизации: sync all – барьерная синхронизация всех копий; sync images – барьерная синхронизация группы;
- операторы блокировки lock и unlock управляют доступом к данным;
- конструкция critical . . . end critical создает критическую секцию для выполнения кода только одной копией;

Компиляция программы с комассивами при помощи компилятора ifort осуществляется с

4 Нумерация экземпляров программы идёт с 1.

использованием команды
ifort -coarray -coarray-num-images=n file.f90 -o file
здесь n – количество копий программы, file – имя исполняемого файла, file.f90 – имя файла с исходным кодом.
К сожалению, на сегодняшний день комассивы полноценно не поддерживаются в свободном компиляторе gfortran.
Ниже приведён код программы численного интегрирования с использованием технологии комассивов:

```
program integral_l
  real(8), codimension[*] :: integral
  integer :: i,x1,y1,x2,y2
  real(8) :: a,b,h,s, n,x
  a=0.0
  b=1.0
  n=1000000000
  h=(b-a)/n
  s=0.0
  sync all
  call system_clock(x1,y1)
!Вычисление частичной суммы  в текущей копии программы.
  do i= this_image(),n,num_images()
    x=a+h*(i-1)
    s=s+4/(1+x*x)
  enddo
  integral=s*h
  sync all
  if (this_image() == 1) then
!Сборка интеграла в первой копии
    do i=2,num_images()
      integral=integral+integral[i]
    enddo
  end if
  call system_clock(x2,y2)
  if (this_image() == 1) then
 write(*, *) 'Интеграл=' , integral , 'Время=', real(x2)/y2-real(x1)/y1, 'Число копий=', num_images()
  endif
end program
```

Ниже приведён код параллельной программы простейшего ленточного алгоритма умножения матриц с использованием технологии комассивов.

```
 real(8) :: res[*]
  real(8),allocatable:: c(:,:)[:]
  real(8),allocatable :: a(:,:),b(:,:)
  integer :: i,j,kn,x1,y1,x2,y2,r,p
  n=2000
  allocate(a(n,n),b(n,n),c(n,n)[*])
  a=1
  b=1
  do i=1,n
  a(i,i)=3
    b(i,i)=2
  end do
  sync all
  call system_clock(x1,y1)
  r=this_image()
  p=num_images()
  jn=(r-1)*n/p+1
  jk=r*n/p
! Формирование полосы матрицы в каждой из копий программы.
! В связи с тем, что в Фортране матрицы хранятся по столбцам, то разбиваем на
```

```
!полосы матрицу B (jn-начало полосы, jk — конец полосы)
!и формируем полосам матрицу C одним оператором фортрана
  c(:,jn:jk)=matmul(a,b(:,jn:jk))
   sync all
  if (this_image() == 1) then
! Сборка сформированных лент матрицы в первой копии
  do r=2,p
  jn=(r-1)*n/p+1
  jk=r*n/p
  C(:,jn:jk)=C(:,jn:jk)[r]
  enddo
  call system_clock(x2,y2)
 endif
```

Для разработки параллельных программ в Фортране поддерживаются все известные технологии: MPI, OpenMP, многопоточное программирование, а также специальный тип массивов Фотрана – комассивы.

Как видно из приведенных выше кодов, использование комассивов и языка Фортран делает написание параллельных программ более простым и доступным не только ИТ-специалисту, работающему в области параллельного программирования, но и профессионалам из различных отраслей при возникновении необходимости написания параллельного приложения.

При решении вычислительных задач исследователю необходимо иметь возможность построения графиков. Используя язык Фортран, разработчик вычислительной программы имеет несколько вариантов построения графиков:

1. Свободная библиотека OpenGL (www.opengl.org) позволяет самостоятельно разрабатывать кроссплатформенное графическое приложение любой сложности.
2. Свободная кросс-платформменная библиотека высококачественной научной графики MathGL (http://mathgl.sourceforge.net/doc_ru/Main.html). Её использование позволит построить двух- или трёхмерный график с помощью нескольких операторов. Синтаксис функций, используемых в MathGL, подобен синтаксису MathLab, Octave, Scilab, GnuPlot [7].
3. Свободная библиотека для построения графиков Plplot (http://plplot.sourceforge.net)
4. Проприетарная бесплатная библиотека высококачественной графики Dislin (http://www.mps.mpg.de/dislin), разработанная в Институте Макса Планка.
5. Свободный модуль gnufor2 (http://www.math.yorku.ca/~akuznets/gnufor2/), который предоставляет программный интерфейс для доступа к свободному кроссплатфоменному приложению построения высококачественных двух- и трёхмерных графиков Gnuplot (http://www.gnuplot.info).

Рассмотренные выше возможности языка Фортран позволяют сделать следующие выводы:

1. Современный язык программирования поддерживает все технологии программирования, встречающиеся при решении вычислительных задач, которые возникают в технике, экономике, медицине и др. остраслях.
2. Код на Фортране хорошо читаем.
3. За более чем полувековую историю накоплено большое количества кода. Многие современные компиляторы поддерживают синтаксис Fortran-66, Fortran-IV. Это позволяет воспользоваться всем опытом человечества в области вычислительной математики.
4. Язык прост в освоении, написание программ на Фортране не сложнее, чем в системах компьютерной математики таких, как Matlab, Scilab, Octave.
5. Для Фортрана существует большое количество компиляторов как свободных, так и проприетарных. С помощью этих компиляторов можно создавать высокоэффективные кроссплатформенные приложения. Создаваемые с помощью компиляторов Фортрана приложения, являются одними из самых быстрых.
6. В Фортран поддержка технологий параллельного программирования внесена на уровне стандарта языка. Поддерживаются конвейерные операции, параллельные циклы и комассивы.
7. При программировании на Фортране поддерживаются графические библиотеки, что позволяет в вычислительные программы включать графический вывод результатов.

Всё выше перечисленное позволяет рекомендовать не только применять Фортран в научных исследованиях, но и рекомендовать использовать его в качестве языка программирования в курсах информатики и программирования для студентов инженерного и физико-математического направлений. В связи с большими возможностями языка при разработке

параллельных приложений имеет смысл знакомить с ним и студентов ИТ-направлений в курсах, связанных с параллельными вычислениями.

Литература

1. Алексеев Е.Р., Шмакова М.С. Язык программирования Фортран: история развития и современность. ОБЩЕСТВО, НАУКА, ИННОВАЦИИ. (НПК – 2015): Всерос. ежегод. науч.-практ. конф.: сб. материалов, 13–24 апреля 2015 г. / Вят. гос. ун-т. – Киров, 2015. – 1 электрон. опт. диск (CD-ROM). –С.1540-1544.
2. Шмакова М.С. Эволюция численных методов и развитие языка Фортран: ТПЖА.010441.109 ПЗ: Дипломная работа / ВятГУ, каф. ПМиИ; рук. Е.Р. Алексеев. - Киров, 2015. - ПЗ 81 с., 22 рис., 13 табл., 115 источников, 4 прил.
3. Горелик А.М. Программирование на современном Фортране. — М.: Финансы и статистика, 2000. – 450 с.
4. Звягин В.Ф., Фёдоров С.В. Параллельные вычисления в оптике и оптоинформатике: Учебное пособие. — СПб: СПбГУ ИТМО, 2009. – 109 с.
5. Артёмов И.Л. Современный Фортран: основы программирования. — М.:Диалог-МИФИ, 2007. – 304 с.
6. Бартеньев О.В. Современный Фортран. — М.: Диалог-МИФИ, 2000. – 450 с.
7. Е.Р.Алесеев, Г.Г.Злобин, Д.А.Костюк,О.В.Чеснокова, А.С.Чмыхало. Программирование на языка C++ в среде Qt Creator. – М.: ALT Linux, 2005, – 448с.

References

1. Alekseev E.R., Shmakova M.S. Yazyk programmirovaniya Fortran: istoriya razvitiya i sovremennost. OBSCCHESTVO, NAUKA, INNOVATSII. (NPK – 2015): Vseros. edzegod. nauch.-prakt. konf.: sb. materialov, 13–24 aprelya 2015 g. / Vyat. gos.un-t. – Kirov, 2015. – 1 elektron. opt. disk (CD-ROM). –S.1540-1544.
2. Shmakova M.S. Evolyutsiya chislennix metodov i razvitie yazika Fotran: TPDZA.010441.109 ПЗ: Diplomnaya / VytaGU, kaf. PMiI; ruk. E.R. Alekseev. - Kirov, 2015. - PZ 81 s., 22 ris., 13 tabls., 115 istochnikov, 4 pril.
3. Gorelik A.M. Programmirovanie na sovremennom Fortrane. — M.: Finansi I statistika, 2000. – 450 s.
4. Zvyagin V.F., Fyodorov S.V. Parallel'nie vichisleniya v optike i optikoinformatike: Uchebnoe posobie. — SPb: SPbGU ITMO, 2009. – 109 s.
5. Artyomov I.L. Sovremeniy Fortran: osnovi programmirovaniya. — M.:Dialog-MIFI, 2007. – 304 s.
6. Barten'ev O.V. Sovremeniy Fortran. — M.: Dialog-MIFI, 2000. – 450c.
7. E.R.Alekseev, G.G.Zlobin, D.A. Kostyuk, O.V.Chesnokova, A.S.Chmyhalo. Programmirovanie na yazike C++ v srede Qt Creator. – M.: ALT Linux, 2005, – 448s.

Поступила 20.10.2016

Об авторах:

Алексеев Евгений Ростиславович, профессор кафедры фундаментальной информатики и прикладной математики Вятского государственного университета, кандидат технических наук, доцент, er.alekseev@yandex.ru;

Соболева Ольга Вячеславовна, программист ООО «Студия «Атум», soboleva-olgav@yandex.ru**.**