ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 130 за 2018 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Горелик А.М.

Эволюция языка Фортран. Устаревшие черты языка и средства для их замены

Рекомендуемая форма библиографической ссылки: Горелик А.М. Эволюция языка Фортран. Устаревшие черты языка и средства для их замены // Препринты ИПМ им. М.В.Келдыша. 2018. № 130. 13 с. doi:10.20948/prepr-2018-130
URL: http://library.keldysh.ru/preprint.asp?id=2018-130

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

А.М.Горелик

Эволюция языка Фортран. Устаревшие черты языка и средства для их замены

Москва — 2018

Горелик А.М.

Эволюция языка Фортран.
Устаревшие черты языка и средства для их замены

Представлены новые возможности современного Фортрана. Рассматривается концепция эволюционного развития языка. Приводятся списки удаленных из стандарта средств языка и перечень средств, признанных устаревшими, но которые пока еще сохранены в языке. Даны методические рекомендации по замене этих средств современными элементами языка.

Ключевые слова: современный Фортран, стандарты Фортрана, эволюция языка Фортран.

Alla Moiseevna Gorelik

Evolution of the Fortran language.
Decremental features of language and means for their replacement

The principal new features of modern Fortran are listed. Evolutional conception of Fortran development is considered. The obsolescent and deleted features are listed. Rationale concerning replacement the features by modern Fortran elements are given.

***Key words*:** modern Fortran, Fortran standards, evolution of the Fortran language.

Оглавление

1. Введение

В 2017 году исполнилось 60 лет языку программирования Фортран. Но язык не умирает. Он постоянно развивается и совершенствуется в соответствии с развитием вычислительной техники, языков и технологий программирования. Фортран все еще актуален при решении сложных вычислительных задач, особенно при использовании современных суперкомпьютеров.

Разработаны и реализованы международные стандарты языка**:** Фортран 66, Фортран 77, Фортран 90, Фортран 95, Фортран 2003 и Фортран 2008 [1–7]. Практически завершена разработка нового стандарта – Фортран 2018 (прежнее рабочее название – Фортран 2015) [8]. Начата разработка проекта следующего стандарта, рабочее название которого – Фортран 202x.

Стандартизация языка создает предпосылки для повышения мобильности программного обеспечения. Современные стандарты Фортрана позволяют использовать новые технологии программирования, что, в свою очередь, позволяет уменьшить время и трудоемкость разработки сложных программ для решения задач, требующих большого объема вычислений на компьютерах различной архитектуры.

Официальные описания стандартов весьма объемные, строго формализованы и сложны для массового пользователя. Неформальные описания и анализ средств стандартов, а также методики их эффективного использования приведены в [9–18].

Данная статья продолжает серию публикаций автора, посвященных анализу различных аспектов современного Фортрана.

Некоторые конструкции языка Фортран в настоящее время устарели и стали излишними после введения новых элементов.

В статье приводится перечень устаревших элементов языка и даются рекомендации по их замене современными средствами.

2. О современном Фортране

Как отмечалось во введении, новшества Фортрана ориентированы на современную архитектуру и современные технологии программирования, что обеспечивает современный стиль программирования и позволяет создавать мобильные, надежные и хорошо структурированные программы, а потому наглядные и лаконичные.

Ниже перечислены некоторые сильные стороны современного Фортрана.

1. Векторные операции (операции над целыми массивами и секциями массивов на поэлементной основе).
2. Структуры данных (производные типы).
3. Средства параметризации типов данных.
4. Управляющие структурные конструкции (if – end if, do – end do, select case – end select).
5. Операции и присваивания, определяемые в программе.

6. Различные механизмы динамического размещения объектов в памяти.
7. Указатели.
8. Расширенные средства ввода/вывода
9. Расширенный набор средств для процедур (внутренние и внешние процедуры, модульные процедуры, рекурсивные процедуры, ключевые и необязательные аргументы, процедуры без побочного эффекта, поэлементные процедуры, явный интерфейс и др.)
10. Средства декомпозиции программ (модули, подмодули).
11. Объектно-ориентированное программирование (наследование, классы, инкапсуляция, полиморфизм, перегрузка операций и процедур).
12. Средства взаимодействия с Си.
13. Большой набор встроенных процедур (в т.ч. поэлементные, справочные, функции редукции, часто используемые математические функции и др.).
14. Средства поддержки параллельных вычислений (coarrays).
15. Концепция эволюционного развития языка (см. следующий раздел).

Средства 1–7 введены в Фортране 90 [4, 9], элементы 8, 9 введены в Фортране 90/95 [4, 3, 9].

Средства 10 – модули введены в Фортране 90 [4, 9], подмодули – в Фортране 2008 [1], черты 11 частично появились в Фортране 90 [4, 9] и в полном объеме – в Фортране 2003 [2, 16]. Средства 12 первоначально введены в Фортране 2003 [2, 18].

Средства 13 первоначально введены в Фортране 90 [4, 9] и получили существенное развитие во всех последующих стандартах [1– 4, 12, 15]. **Средства поддержки параллельных вычислений Coarray введены в Фортране 2008** [1, 17]**.**

3. Концепция эволюционного развития языка

Каждый последующий стандарт расширяет возможности своих предшественников. В то же время некоторые конструкции языка Фортран в настоящее время устарели и стали излишними после введения новых элементов.

Наличие в языке устаревших черт усложняет язык и компилятор, что негативно отражается на эффективности изготовляемых программ. Кроме того, некоторые устаревшие элементы являются источником ненадежности, другие – влияют на снижение мобильности, многие из таких черт противоречат концепции структурного программирования и затрудняют распараллеливание.

Однако исключение из языка каких-либо элементов при очередном пересмотре стандарта может быть весьма болезненным, **оно может привести к тому, что будет затруднительно использовать существующий фонд программного обеспечения на Фортране.**

Вместе с тем ясно, что, если в язык будут добавляться все новые и новые средства без удаления уже ненужных, устаревших элементов, язык станет очень громоздким с большим числом дублирующих элементов.

Учитывая эти противоречивые факторы, разработчики стандарта приняли следующее решение: элементы языка, которые являются кандидатами на удаление, заранее объявляются устаревшими, нерекомендуемыми для использования**;** в каждом стандарте языка Фортран приводятся два списка устаревших черт с различным статусом.

Первый список содержит удаляемые (deleted) из данного стандарта черты – те элементы языка, которые в предыдущем стандарте были объявлены как отживающие и нерекомендуемые и теперь удаляются.

Второй список содержит отживающие (obsolescent) и нерекомендуемые для использования черты – те элементы языка, которые пока сохраняются для преемственности, но являются кандидатами на удаление в следующей ревизии стандарта (хотя они необязательно должны быть удалены).

Программист должен избегать использования этих элементов при написании новых программ и при модификации "старых". Стандартные модули, предназначенные для различных приложений, как средство расширения языка, не должны содержать устаревших черт.

Устаревшие черты могут оставаться в реализации, однако современные стандарты Фортрана требуют, чтобы компилятор выдавал сообщение по поводу тех конструкций, используемых в программе, которые являются устаревшими или исключенными.

Ниже в разделе 4 приводятся списки устаревших и удаленных из стандартов (Фортран 90 – Фортран 2008) элементов языка и рекомендации по их замене современными средствами. В разделе 5 рассматриваются те черты языка Фортран, которые планируется включить в списки устаревших и удаленных черт в проекте будущего стандарта языка, рабочее название которого – Фортран 2018. В разделе 6 анализируются те черты стандарта, которые устарели, но пока еще не включены в списки устаревших черт.

4. Удаленные и устаревшие черты в стандартах языка Фортран и средства для их замены

4.1. Устаревшие черты

Устаревшие черты для стандарта – это те средства, которые хотя и признаны устаревшими, были сохранены в языке для преемственности с предыдущими стандартами.

Список устаревших элементов.

1. Операторные функции (statement functions).
2. Оператор go to со списком меток (computed go to statement).
3. Альтернативный возврат из подпрограммы.

4. Операторы data среди выполняемых операторов.
5. Функции символьного типа, перенимающие длину.
6. Фиксированный формат исходной программы.
7. Операторы entry.
8. Форма оператора character со звездочкой.
9. Общий оператор окончания вложенных циклов и использование в качестве последнего оператора цикла какого-либо оператора отличного от continue и end do.

Каждое из перечисленных средств может быть эффективно заменено более современными средствами. Рассмотрим их более детально.

1) Операторные функции (statement functions).

Операторная функция является источником потенциальных ошибок, так как синтаксис объявления такой функции похож (и потому легко может быть перепутан) на синтаксис оператора присваивания.

Вместо операторных функций лучше использовать внутренние процедуры; такие процедуры имеют более общую форму, чем операторные функции, и полностью покрывают их возможности.

2) Оператор перехода со списком меток (computed go to statement).

Этот оператор обеспечивает выбор одной из ветвей программы в зависимости от значения некоторого выражения.

Та же возможность более эффективно обеспечивается с помощью конструкции выбора select case – end select, которая больше соответствует современному стилю программирования.

3) Альтернативный возврат из подпрограммы.

Для вызова процедуры, содержащей альтернативный возврат, в список аргументов помещаются метки, что позволяет вызванной процедуре после возврата управлять выполнением вызывающей процедуры. Эта возможность противоречит концепции структурного программирования.

Того же эффекта можно достичь, используя код возврата в конструкции выбора после возврата в вызывающую процедуру. Конструкция выбора больше соответствует современному стилю программирования.

4) Операторы data среди выполняемых операторов.

Эта возможность редко используется. Инициализацию данных разумнее помещать среди операторов спецификации. При необходимости можно использовать конструкцию block – end block, которая позволяет помещать операторы объявления среди выполняемых операторов.

5) Символьные функции, перенимающие длину (assumed character length functions).

Такую функцию лучше заменить подпрограммой (subroutine), аргументы которой соответствуют результату функции и ее аргументам.

6) Фиксированный формат исходной программы.

Разработчики первых вариантов Фортрана руководствовались тем, что ввод программы осуществлялся только с перфокарт, и поэтому нетрудно было выполнить требование, чтобы символы исходной программы располагались на определенных позициях строки перфокарты. Так был принят фиксированный формат записи программы, который впоследствии оказался неудобным при использовании терминалов. Действительно, при вводе программы с клавиатуры неудобно начинать операторы с седьмой колонки, а шестую использовать для признака продолжения.

Введенный в язык свободный формат более технологичен, не требуется размещать программу в определенных позициях.

7) Форма оператора character со звездочкой.

В языке имеются две альтернативные формы для спецификации длины символьных данных: старая форма (со звездочкой) и новая форма (без звездочки), которая более естественна.

8) Операторы entry.

Оператор entry позволяет задавать дополнительные входы в процедуру с альтернативным списком аргументов. Это усложняет язык и противоречит концепции структурного программирования.

Необходимость в этом операторе отпадает с появлением программных единиц-модулей, в которых аналогичные возможности обеспечиваются с помощью модульных процедур.

9) Общий оператор окончания вложенных циклов и использование в качестве последнего оператора цикла какого-либо оператора отличного от continue и end do.

Следует использовать оператор end do или continue для каждого оператора do, что больше соответствует принципам структурного программирования.

4.2. Удаленные черты

Перечисленные ниже средства удалены из стандарта и могут быть эффективно заменены более современными средствами языка.

1. Параметры цикла вещественного типа и типа двойной точности.
2. Передача управления конечному оператору (end if) извне конструкции if – end if.
3. Оператор паузы (pause).
4. Оператор присваивания метки (assign) и оператор go to по предписанию.
5. Присваивание значения метки оператора format в операторе assign.
6. Дескриптор редактирования h в операторе format.
7. Управление кареткой.

Ниже даются рекомендации по замене перечисленных средств.

1) Параметры цикла вещественного типа и типа двойной точности.

В заголовке цикла допускались (наряду с целым типом) параметры вещественного типа и типа двойной точности.

Рекомендуется их заменить на параметры целого типа, или лучше заменить на конструкцию цикла do – end do, в которой отсутствуют параметры цикла.

2) Передача управления конечному оператору (end if) извне конструкции if – end if.

Такая передача управления не имеет смысла. Вместо этого следует передавать управление оператору, непосредственно следующему за end if.

3) Оператор pause.

Выполнение оператора pause требовало операторского или системного вмешательства для возобновления исполнения. В большинстве случаев этой же возможности (но более эффективно и более мобильным способом) можно достичь путем использования подходящего оператора read, ожидающего ввода некоторых данных.

4) Оператор присваивания метки assign и оператор go to по предписанию.

Оператор assign позволял динамически присваивать метку переменной целого типа, а оператор go to по предписанию обеспечивал "непрямое ветвление" с использованием этой переменной. Это ухудшает наглядность программы, особенно если переменная используется также в числовых операциях. Два различных способа использования переменной могут стать источником ошибки. Эти операторы обычно использовались для моделирования внутренних процедур, которые теперь введены в Фортран.

5) Присваивание значения метки оператора format в операторе assign. Оператор assign позволял также динамически присваивать метку оператора format переменной целого типа, которую затем можно использовать как спецификатор формата в операторах read, write или print. Это ухудшает наглядность, допускает несогласованное использование целой переменной и может быть источником ошибки.

В этом случае альтернативой оператору assign может служить символьное выражение для спецификации формата.

6) Дескриптор редактирования h в операторе format.

При использовании этого дескриптора редактирования необходимо было подсчитывать количество символов, это неудобно, и кроме того, использование такого дескриптора может быть источником ошибки.

Та же возможность достижима при использовании дескриптора редактирования “символьная строка”, при этом не требуется подсчитывать количество символов.

7) Управление кареткой.

Операторы вывода в Фортране первоначально были ориентированы на печатающие устройства, понятиями для которых являются строки и страницы печати**;** первый символ каждой записи не печатается, он рассматривался как символ управления кареткой. Сейчас это удалено.

5. Новые удаленные и устаревшие черты

В проекте будущего стандарта языка (в Фортране 2018) предусмотрены новые списки устаревших и удаленных элементов.

5.1. Новые удаленные черты

1. Арифметический оператор if.
2. Неблочные конструкции цикла.

Рекомендации по замене.

1) Арифметический оператор if.

Оператор арифметический if содержит три метки и делает код не наглядным, препятствует оптимизации и снижает мобильность.

Этот оператор ранее был объявлен как устаревший, теперь удаляется из стандарта. Целесообразно заменить его на оператор if логический или на конструкцию if – end if. Оба варианта, в отличие от арифметического if, обеспечивают структурный стиль программирования.

2) Неблочные конструкции цикла.

Общий оператор окончания вложенных циклов и использование в качестве последнего оператора цикла какого-либо оператора отличного от continue и end do.

Ранее эта черта была признана устаревшей, теперь удаляется из стандарта.

Следует использовать оператор end do или continue для каждого оператора do, что больше соответствует принципам структурного программирования.

Заметим, что оператор do с меткой (с указанным ограничением) пока сохраняется в языке, но признан устаревшим.

5.2. Новые устаревшие черты

1. Оператор common.
2. Оператор equivalence**.**
3. Программная единица блок данных (block data).
4**.** Оператор do с меткой.
5. Оператор и конструкция forall.
6. Специфические имена встроенных функций.

Рекомендации по замене новыми средствами языка.

1) Оператор common.

Введенный в Фортран новый вид программных единиц-модулей обеспечивает гораздо более мощные средства, чем общие блоки, определяемые с помощью операторов common. Основное преимущество модулей заключается в том, что они предоставляют пользователю возможность работать не только с общими данными, но также с другими общими объектами – типами данных,

заготовленными описаниями процедур и определяемых операций. Способ работы с глобальными данными, обеспечиваемый модулями, более гибкий, чем с помощью операторов common: глобальные данные, описанные в модуле, доступны по имени, а не по положению внутри общего блока. Кроме того, в отличие от описания общих блоков в операторах common, глобальные переменные в современном Фортране надо описывать только в одном месте.

2) Оператор equivalence.

Оператор equivalence обеспечивает совместное использование памяти несколькими объектами данных; такая возможность используется обычно либо для экономии памяти (что было важно, когда размер памяти был недостаточным), либо для моделирования структуры данных с компонентами разных типов. Однако, во-первых, этот искусственный прием не удобен, и, во-вторых, оператор equivalence имеет ряд ограничений. Кроме того, хранение в одном и том же месте памяти данных разного типа делает программу менее надежной и усложняет ее модификацию.

Использование эквивалентности по памяти вообще противоречит принципам структурного и надежного программирования.

В разных ситуациях, где используется эквивалентность, можно вместо нее использовать производные типы, указатели, динамически размещаемые массивы или функцию transfer.

3) Программная единица блок данных (block data).

Программные единицы block data существуют только для того, чтобы инициализировать данные в common блоках. Использование программных единиц-модулей и средств инициализации в операторах объявления типа устраняет необходимость в использовании программных единиц блоков данных.

4) Оператор do с меткой.

Метка в операторе цикла избыточна после введения в язык имени конструкции и с использованием оператора cycle, который нагляднее.

5) Оператор и конструкция forall.

Этот оператор избыточен и имеет много ограничений. Лучше использовать оператор параллельный цикл (do concurrent).

6) Специфические имена встроенных функций.

Специфические имена встроенных функций теперь избыточны и признаны устаревшими, они редко используются и препятствуют мобильности. Все встроенные функции имеют универсальные имена. Рекомендуется использовать универсальные имена.

6. Другие нерекомендуемые черты

Ниже приводится список черт, которые пока еще не включены в список устаревших, но их не рекомендуется использовать, так как имеются современные элементы для реализации тех же возможностей. Приводятся

предложения по замене этих черт современными элементами, уже введенными в Фортран.

1) Вещественный тип двойной точности (double precision).

В предыдущих стандартах предусматривались две разновидности вещественного типа (real и double precision), которые позволяли задавать только относительную длину представления данных. Это приводило к некоторым проблемам при переносе программы из одной компьютерной системы в другую, так как при переносе подчас требовалось модифицировать программу так, чтобы использовать данные двойной точности вместо обычных вещественных. В современных стандартах Фортрана имеются средства для спецификации точности и диапазона (параметры типа), которые больше соответствуют требованиям мобильности, позволяют компилятору самому определить способ отображения на конкретную архитектуру. Вещественный тип двойной точности целесообразно заменить вещественным типом с соответствующим параметром разновидности типа.

Пример

```
real (selected_real_kind (10, 30)) x, y
```

Этот оператор обеспечивает точность не менее 10 знаков и диапазон десятичного порядка не менее чем (-30, +30).

2) Связь скаляра и массива при передаче аргументов процедур.

Стандарт разрешает, чтобы формальному аргументу-массиву соответствовал фактический аргумент – элемент массива. При этом подразумевается, что с формальным аргументом связывается данный фактический элемент массива и все элементы, следующие за ним в программе в обычном для массива порядке. Такая связь аргументов приводит к плохо структурированной программе, ухудшает ясность и снижает эффективность.

В языке имеются более эффективные средства, обеспечивающие передачу части массива – секции массива.

3) Массивы, перенимающие размер.

Напомним, что массивы, перенимающие размер, – это формальные аргументы, в описании которых верхняя граница последнего измерения задается звездочкой; такие массивы наследуют размеры соответствующих фактических аргументов.

В качестве обобщения понятий массивов, перенимающих размер, в Фортран введены массивы, перенимающие конфигурацию, т.е. массивы (формальные аргументы), которые наследуют не только размер, но и конфигурацию соответствующих фактических аргументов. В описании таких массивов опущены верхние границы. Использование массивов, перенимающих конфигурацию, обеспечивает больше возможностей по сравнению с использованием массивов, перенимающих размер.

4) Строка include.

Строка include обеспечивает возможность подключать к исходному тексту какой-нибудь другой программный текст.

Тот же эффект с большим разнообразием предоставляемых возможностей обеспечивается с помощью оператора use; при этом вставляемый текст помещается в программной единице-модуле.

5) Старые обозначения операций отношения.

Старые обозначения операций отношения .eq., .ne., .lt., .le., .gt., .ge. были введены в язык во времена, когда на клавиатуре отсутствовали необходимые символы.

Новые обозначения ==, /=, <, <=, >, >= более удобны для использования.

7. Заключение

Бытует мнение, что Фортран – устаревший язык. Это ошибочное утверждение. Несмотря на почтенный возраст, современный Фортран – это один из лучших инструментов при решении сложных задач вычислительного характера на современных компьютерах. Приведенный в разделе 2 краткий перечень основных сильных сторон современного Фортрана подтверждает сказанное.

Использование при программировании сложных вычислительных задач современных средств Фортрана существенно облегчает разработку и отладку программ, упрощает распараллеливание и позволяет создавать более конкурентоспособный продукт по сравнению с использованием “старого” Фортрана.

С введением новых элементов языка некоторые средства стали избыточными. Однако удаление устаревших элементов происходит постепенно с большой осторожностью, с тем чтобы не возникало проблем, связанных с использованием ранее написанных программ (см. раздел 3). В статье приведены рекомендации по эффективной замене устаревших и удаленных средств современными элементами языка (см. раздел 4).

Развитие Фортрана продолжается. В проекте будущего стандарта языка (в Фортране 2018) получили дальнейшее развитие средства взаимодействия Фортрана и Си, дополнительные средства поддержки параллельности (расширения средств coarrays), расширяются списки устаревших и удаленных черт (см. раздел 5) и вводятся другие новшества.

Современный Фортран реализован на различных платформах и используется при решении задач вычислительного характера как в ИПМ, так и в других организациях в России и за рубежом. Развиваются системы параллельного программирования, ориентированные на современный Фортран: MPI, OpenMP, Coarray, DVM и др.

Литература

1. ISO/IEC 1539-1: 2010. *Information technology – Programming languages – Fortran – Part1: Base Language.*
2. ISO/IEC 1539-1: 2004. *Information technology – Programming languages – Fortran – Part1: Base Language.*
3. ISO/IEC 1539-1: 1997. *Information technology – Programming languages – Fortran – Part1: Base Language.*
4. ISO/IEC 1539: 1991(E). *Information technology – Programming languages – Fortran.*
5. ANSI X3.9-1966. *USA Standard FORTRAN.*
6. ANSI X3.9-1978. *American National Standard – Programming Language FORTRAN* (ISO 1539-1980).
7. Фортран 90. Международный стандарт. Перевод с англ. Дробышевич С.Г. редактор перевода Горелик А.М. М.: Финансы и статистика, 1998. С.378.
8. Reid J. The new features of Fortran 2015. ACM SIGPLAN Fortran Forum, 2017, №2, pp.3-38.
9. Горелик А.М. Программирование на современном Фортране. – М.: Финансы и статистика, 2006. 351 с.
10. Горелик А.М. Новый стандарт языка Фортран (Фортран 2008) // Препринты ИПМ им.М.В.Келдыша. 2011. № 16. С. 1-29.
11. Горелик А.М. Эволюция языка программирования Фортран (1957–2007) и перспективы его развития // Вычислительные методы и программирование, 2008, т.9, №2. С.223-241.
12. Gorelik A.M. Statements, data types and intrinsic procedures in the Fortran Standards // ACM SIGPLAN Fortran Forum, 2014, №3, pp. 5-17.
13. Горелик А.М. Современные международные стандарты языка Фортран. // Программирование. 2001, № 6, С. 44-56.
14. Горелик А.М. Современный Фортран для компьютеров традиционной архитектуры и для параллельных вычислительных систем. // Вычислительные методы и программирование. М.: МГУ. 2004. Т.5. С. 320-328.
15. Горелик А.М. Сравнение стандартов языка Фортран // Информационные технологии и вычислительные системы, 2015. №3. С. 45- 64.
16. Gorelik A.M. Object-Oriented Programming in Modern Fortran // Programming and Computer Software, vol.30, №3, 2004, pp. 173-179.
17. Горелик А.М. Средства поддержки параллельных вычислений в стандартах языка Фортран. // Информационные технологии и вычислительные системы, 2014, № 3. С. 17-23.
18. Горелик А.М. Смешанное программирование на Фортране и Си // Информационные технологии и вычислительные системы, 2016, № 3. С. 62-67.