УДК 004.43

ЭВОЛЮЦИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ ФОРТРАН (1957 – 2007) И ПЕРСПЕКТИВЫ ЕГО РАЗВИТИЯ

А. М. Горелик[1]

В статье дается исторический обзор развития языка программирования Фортран. Рассматривается эволюция основных средств языка и новшества, введенные в современные международные стандарты. Приведены таблицы отличий основных элементов языка в различных стандартах (от Фортрана 66 до Фортрана 2003). Приводится информация о реализациях Фортрана в СССР и о перспективах развития языка. Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 06–01–00574).

1. Введение. Язык программирования Фортран перешагнул 50-летний рубеж. В прошлом году исполнилось 50 лет первому Фортран-компилятору. Это событие освещалось в публикациях и в Интернете. Так, журнал Scientific Programming посвятил юбилею специальный выпуск (Vol. 15, N 1, 2007). Полувековой юбилей — время проследить путь развития и заглянуть в будущее языка Фортран. Несмотря на то что в последующие годы появились новые языки, Фортран по-прежнему занимает лидирующие позиции при решении сложных вычислительных задач.

Конечно, язык серьезно изменился. При разработке современного Фортрана (Фортран 90/95/2003) учтены новые технологии программирования и архитектура современных компьютеров. Современный Фортран позволяет создавать более мобильные и более надежные программы, лучше структурированные, а потому более наглядные и более лаконичные. Язык содержит элементы непроцедурности, средства явной спецификации векторных операций, новые средства декомпозиции программ. Современный Фортран содержит средства поддержки объектно-ориентированного программирования (ООП) и средства параллельного программирования. Разработаны специальные системы параллельного программирования, ориентированные на современный Фортран.

Сохраняется преемственность с предыдущими стандартами языка, что позволяет использовать ранее созданные библиотеки и прикладные программы; новые средства могут быть использованы и для модификации старых программ.

Современный Фортран реализован практически на всех компьютерах и вычислительных системах для различных платформ. Реализованные технологии оптимизации, поддержка параллелизма, автоматическая векторизация позволяют эффективно использовать компиляторы для современной архитектуры. Разработаны интегрированные системы программирования для разработки прикладных программ, графические пакеты, математические библиотеки (в том числе и для параллельной архитектуры) и др.

Работа по развитию и совершенствованию Фортрана продолжается. Разработан и вынесен на дальнейшее обсуждение проект будущего стандарта языка, рабочее название которого — Фортран 2008. Одно из важных новшеств будущего стандарта — средства поддержки параллельных вычислений.

В статье прослеживается путь развития языка от его появления до сегодняшних дней. В разделе 2 дается краткий исторический обзор развития языка; в разделе 3 — информация о реализациях Фортрана в СССР; в разделе 4 — эволюция основных элементов языка; в разделе 5 перечисляются новшества, которые введены в каждый последующий стандарт языка; в разделе 6 обсуждается принятая концепция эволюционного развития языка (устаревшие и удаленные черты). В разделах 7 и 8 приводятся таблицы операторов и встроенных процедур в различных стандартах Фортрана, в разделе 9 перечислены основные новые средства в разрабатываемом проекте будущего стандарта языка.

2. Краткий исторический обзор. Язык Фортран и первый компилятор с языка Фортран разработаны в корпорации IBM коллективом под руководством Дж. Бэкуса. До появления этого языка программы разрабатывались вначале непосредственно в кодах машины, а позднее на языке ассемблера. Язык Фортран быстро завоевал популярность, так как он был простым для изучения и обеспечивал возможность генерации эффективного исполняемого кода. За многие годы разработано большое количество библиотек

[1] Институт прикладной математики им. М.В. Келдыша РАН (ИПМ РАН), Миусская пл., 4, 125047, Москва; e-mail: gorelik@keldysh.ru

и прикладных программ, многие из которых не потеряли свою актуальность. Это является неоспоримым преимуществом языка.

Язык с самого начала был ориентирован на решение научно-технических задач, связанных с обработкой числовой информации. В структуре и основных элементах первых вариантов языка нашли отражение особенности ЭВМ первого поколения, а также используемые в то время технологии программирования.

Постепенно совершенствовалась вычислительная техника, усложнялись задачи, совершенствовались методы программирования. Все это побуждало к развитию языка: так появился сначала Фортран II, а затем Фортран IV. В этих диалектах языка Фортран, с одной стороны, появились новые возможности, а с другой — отброшены некоторые элементы, которые не оправдали себя или устарели. Указанные диалекты языка имели различные варианты, отличавшиеся друг от друга, что существенно снижало мобильность программ, написанных на разных версиях. Это привело к необходимости разработки стандарта языка.

В 1966 году был принят первый американский стандарт языка Фортран [1] для двух вариантов: основного, получившего в дальнейшем название Фортран 66, и подмножества — Базисного Фортрана. Эти варианты языка примерно соответствовали Фортрану IV и Фортрану II.

С течением времени Фортран 66 тоже перестал удовлетворять требованиям многих пользователей. Разработчики трансляторов стали вносить в языки реализаций различные расширения; появилось много версий Фортрана на базе Фортрана 66, но не совместимых между собой. Возникла необходимость в разработке нового стандарта. Новый вариант Фортрана — Фортран 77 [2] был важным шагом вперед в развитии языка. Фортран 77 закрепил ряд расширений языка, прошедших проверку во многих реализациях, и сохранил в основном преемственность со своим предшественником — Фортраном 66. Среди большого количества нововведений Фортрана 77 отметим средства обработки текстовой информации и средства работы с файлами прямого доступа.

Международная организация по стандартизации ISO сначала в 1972 г., а затем в 1980 г. приняла американские национальные стандарты языка Фортран без изменений: ISO 1539-72 (FORTRAN 66) и ISO 1539-80 (FORTRAN 77). Однако со временем и Фортран 77 перестал отвечать новым требованиям. Основные его недостатки: конструкции языка не позволяли в полной мере применять методы структурного программирования и отсутствовали средства работы с динамическими массивами.

После утверждения стандарта Фортрана 77 началась работа по разработке нового международного стандарта, который получил сначала условное название Фортран 8x, а затем — Фортран 90 (Fortran 90) [3, 4]. Фортран 90 (принятый в 1991 г.) открыл новый этап в развитии языка. История разработки проекта стандарта языка Фортран 90 была не простой. В отличие от стандарта Фортрана 77, в стандарте Фортрана 90 не только закреплены появившиеся в различных реализациях расширения, но и появились совершенно новые элементы. Введены средства, которые облегчают разработку программ и позволяют использовать новые технологии программирования; впервые введены некоторые средства поддержки параллельности. Фортран 90 также сохранил преемственность со своим предшественником — Фортраном 77.

В 1997 г. был принят новый международный стандарт языка — Фортран 95 (Fortran 95) [5]. По сравнению с Фортраном 90, в Фортране 95 новшеств сравнительно немного: в основном они связаны с параллельностью и ориентированы на использование HPF (High Performance Fortran). Полное описание стандартов Фортран 90/95 на русском языке приводится в книге [6], обзор новых средств — в работе [7].

Следующим важным шагом в развитии языка была разработка стандарта языка Фортран 2003 [8]. Наиболее существенные новшества — полный набор средств поддержки объектно-ориентированного программирования и средства взаимодействия с Си.

3. Фортран в СССР. Фортран в СССР появился позже, чем в других странах, поскольку предпочтение отдавалось языку Алгол 60. Однако в дальнейшем Фортран стал наиболее популярным языком программирования. Разработанные компиляторы (трансляторы) были установлены на большинстве отечественных ЭВМ: “Минск-32” [9], БЭСМ-4, БЭСМ-6 [10 – 13], АС-6 [14], ЕС ЭВМ [15, 16], СМ ЭВМ [17], МВК “Эльбрус” [18] и др. Было выпущено большое количество книг по Фортрану, которые не залеживались на прилавках, язык изучали в вузах.

На БЭСМ-6 было разработано несколько трансляторов. Первым был транслятор Фортран-Дубна [10], разработанный в ОИЯИ (Дубна). Реализованная версия основана на Фортране 66. Транслятор широко использовался в различных организациях.

В ИПМ им. М.В. Келдыша АН СССР в разное время было разработано несколько трансляторов. Два из них, Фортран-Алмо и Форшаг (Фортран шаговый) [11, 12], были написаны на языке Алмо [19] и генерировали программу на языке Алмо. Это позволило автоматически установить трансляторы на несколько разных типах ЭВМ. В обоих трансляторах реализован стандарт языка Фортран 66. Форшаг, кроме того,

включал язык диалога, использование которого позволило составлять, редактировать и транслировать программы в диалоговом режиме. Транслятор Форшаг был установлен и использовался на БЭСМ-6, ЭВМ "Весна" и на гибридно-вычислительной системе ГВС-100.

В том же Институте позже был разработан транслятор Форекс (Фортран расширенный), который использовался на БЭСМ-6 [13] и АС-6 [14]. Форекс является расширением Фортрана 66, реализованы многие черты Фортрана 77. В трансляторе заложен мощный аппарат оптимизации, который позволяет генерировать эффективный код. Разработан также комплекс графических программ для Фортрана — Графор [20], который широко использовался на различных компьютерах.

Сравнение эффективности программ, генерируемых разными трансляторами для БЭСМ-6, приведено в работе [21]. Использовались варианты трансляторов с оптимизацией и без оптимизации. Сравнение проводилось как на реальных прикладных задачах, так и на специальных тестах. Исследование показало, что время выполнения для многих тестов существенно зависит от транслятора и уровня оптимизации, в то время как на реальных прикладных задачах это различие не велико.

Для ЕС ЭВМ [15, 16] на разных этапах имелось несколько трансляторов, которые базировались на стандартах Фортран 66 и Фортран 77 с расширениями и были ориентированы на разные операционные системы этих ЭВМ.

Реализованные в разных трансляторах версии языков отличались друг от друга, и разработчики прикладных программ, которые ориентировались на конкретный транслятор, часто попадали в затруднительное положение при переносе программы в другое окружение (даже если использовался один и тот же тип компьютера). Это привело к осознанию необходимости проведения исследований по проблемам мобильности и переносу программ. В работе [22] подведены итоги этих исследований; в ней содержится детальный перечень отличий десяти версий Фортрана, реализованных на отечественных ЭВМ, приводятся методики написания мобильных программ и рекомендации по переносу Фортран-программ из одного окружения в другое. К этой работе был проявлен интерес и за рубежом, авторам было предложено принять участие в обсуждении проекта международного стандарта Фортран 8x (впоследствии Фортран 90) и в дальнейшем войти в состав рабочей группы по стандартизации Фортрана (ISO/IEC JTC1/SC22/WG5).

4. Эволюция основных элементов языка Фортран. В данном разделе описывается эволюция основных традиционных элементов языка Фортран — формата исходной программы, структур и типов данных, средств работы с массивами, управляющих конструкций, процедур, средств ввода-вывода, средств декомпозиции программ и средств поддержки параллельности.

4.1. Формат исходной программы и имена. В первых вариантах Фортрана был предусмотрен фиксированный формат записи программы, при котором символы исходной программы располагались на определенных позициях строки-перфокарты. Впоследствии фиксированный формат оказался неудобным при использовании терминалов. В стандартах, начиная с Фортрана 90, наряду с фиксированным форматом исходной программы разрешена более удобная и более наглядная форма — свободный формат. В Фортране 95 фиксированный формат признан устаревшим, хотя и сохранился для преемственности.

В ранних стандартах языка допускались только имена, содержащие до шести символов (букв или цифр), это ограничивало выбор удобных мнемонических обозначений. Начиная с Фортрана 90 имена могут содержать до 31 символа (допускаются буквы, цифры и символ подчеркивания). В Фортране 2008 планируется увеличить допустимую длину до 63 символов.

4.2. Типы, структуры и атрибуты данных. Для Фортрана традиционными всегда были встроенные (предопределенные) простые типы данных (массивы в специальные типы не выделялись). На разных этапах развития Фортрана в язык включалось то или иное множество встроенных типов: сначала только целый и вещественный, затем добавился логический тип, тип двойной точности и комплексный. В ранних вариантах языка не разрешалось смешивать в одном выражении операнды целого и вещественного типов; позже это ограничение было снято.

В Фортране 77 появился символьный тип данных, что явилось одним из наиболее важных новшеств этого варианта языка, так как включение таких данных вместе с соответствующими операциями и функциями позволило использовать Фортран для обработки текстовой информации.

В Фортране 90 введены параметризованные встроенные типы, обеспечивающие средства для задания способа представления данных встроенных типов; расширены возможности для символьного типа данных. Одно из самых важных нововведений — это производные типы (структуры), определяемые в программе, а также определяемые присваивания и определяемые операции над данными любых типов.

В Фортране 77 операторы объявления были организованы так, что для объявления типов и других свойств объектов требовались разные операторы. Современный Фортран позволяет объявлять типы и

другие свойства (атрибуты) объекта в одном операторе.

Для более ранних вариантов языка был характерен принцип умолчания, т.е. автоматическое приписывание объектам типа по первой букве имени (если тип объекта не указан явно). Это не позволяло выполнять некоторые виды контроля на этапе компиляции. В современном языке для отмены автоматического приписывания типа имеется оператор IMPLICIT NONE; при наличии этого оператора типы объектов должны быть объявлены явно.

В Фортране 2003 введены параметризованные производные типы и перечислимые типы. В Фортране 2008 предполагается расширить набор встроенных процедур для работы с битовыми данными.

4.3. Массивы. Средства работы с массивами в процессе развития языка постепенно совершенствовались. В Фортране 66 максимальная размерность массива равнялась трем, нижняя граница всегда предполагалась равной 1, а верхняя граница была строго положительной. В Фортране 77 максимально допустимая размерность массива увеличена до семи; появилась возможность задавать нижние границы измерений массива, причем границы могут иметь и неположительные значения; сняты ограничения на вид индексных выражений.

В стандарте Фортрана 90 и затем в Фортране 95 средства работы с массивами существенно расширены. Введена возможность указывать части массивов (секции), разрешается выполнять операции над целыми массивами и секциями массивов на поэлементной основе, введена конструкция условного присваивания массиву под управлением логической маски (WHERE), введены средства динамического размещения массивов в памяти и большой набор встроенных процедур для работы с массивами.

Новые средства работы с массивами позволяют упростить способ организации и написания почти всех программ, использующих массивы, так как позволяют во многих случаях обойтись без вложенных циклов и условных переходов. Многие алгоритмы, включающие работу с массивами, могут быть записаны более лаконично и наглядно, такие средства улучшают структуру программы; все это облегчает отладку, модификацию и распараллеливание программы.

В Фортране 2003 получили дальнейшее развитие средства работы с динамическими массивами. В Фортране 2008 размерность массивов предполагается увеличить до 15.

4.4. Структурное программирование. Долгое время в Фортране не было адекватных управляющих конструкций для структурного программирования. Арифметический оператор IF, оператор цикла с меткой в конце цикла, операторы предписания ASSIGN, перехода по предписанию GOTO и вычисляемый GOTO (операторы GOTO со списком меток) не удовлетворяют принципам структурного программирования. В Фортране 77 введена только структурная конструкция условия IF ... THEN–ELSE–END IF.

В Фортране 90 упомянутый выше недостаток языка устранен: введены блочные управляющие конструкции, которые поддерживают методологию структурного программирования; операторы, не удовлетворяющие принципам структурного программирования, признаны устаревшими, а в Фортране 95 некоторые из них удалены.

4.5. Процедуры. Процедуры, как и другие элементы языка, претерпели заметные изменения за период существования и развития Фортрана. В Фортране 66 понятие процедуры было уже достаточно развитым: были предусмотрены встроенные функции, операторные функции, внешние процедуры (подпрограммы и функции), определен механизм передачи аргументов (параметров).

В Фортране 77 основные изменения, касающиеся процедур, сводятся к следующему: введена возможность сохранения объектов после завершения выполнения внешней процедуры (оператор SAVE); дополнительные входы в процедуру и альтернативный возврат из процедуры; расширены возможности в связи с введением в язык символьного типа; введены универсальные (родовые) имена встроенных функций; расширен набор встроенных функций.

В Фортране 90 введены новые расширения, связанные с процедурами: внутренние процедуры; рекурсивные процедуры; введено средство спецификации формальных аргументов по характеру их использования (входные, выходные или универсальные); введены ключевые и необязательные аргументы процедур; средства, обеспечивающие возможность описания (с помощью процедур) новых операций и присваиваний; введены средства явного интерфейса с внешними процедурами (интерфейсные блоки). Значительно расширен набор встроенных функций, и введены встроенные подпрограммы.

В последующих стандартах для некоторых встроенных процедур добавлены новые аргументы и расширен набор встроенных процедур.

4.6. Ввод-вывод. Одним из важных преимуществ языка Фортран (уже в самых ранних его вариантах) является богатый набор средств ввода-вывода.

В Фортране 66 были средства ввода с внешних устройств и вывода результатов под управлением различных форматов, а также средства для работы с последовательными файлами.

В Фортране 77 введены средства для работы с файлами прямого доступа, внутренние файлы и новый аппарат управления связью устройств и файлов. Форматы ввода-вывода данных, неплохо развитые и в Фортране 66, получили дальнейшее развитие в Фортране 77.

В Фортране 90 добавлены некоторые новые спецификаторы в операторах ввода-вывода и формата, добавлен ввод/вывод, управляемый списком, и расширен перечень объектов, которые могут использоваться в списках ввода/вывода.

В Фортране 2003 введен асинхронный ввод-вывод и потоковый доступ к файлам. Некоторые новые черты планируются и в Фортране 2008.

4.7. Декомпозиция программ и объектно-ориентированное программирование. Декомпозиция программы предполагает ее разбиение на программные компоненты. Средства декомпозиции особенно полезны для разработки больших программ, так как облегчают разбиение программы на компоненты, которые могут быть разработаны независимо.

Уже в первом стандарте языка были предусмотрены программные единицы — программные компоненты, которые могут транслироваться независимо друг от друга. Начиная с Фортрана 90 появилась возможность использовать более современные средства декомпозиции с помощью нового вида программных единиц — модулей (модули предназначены для спецификации глобальных данных и процедур, для инкапсуляции объектов и др.)

Еще более высокий уровень декомпозиции обеспечивает методология объектно-ориентированного программирования (ООП). В первых вариантах языка практически отсутствовали средства поддержки ООП. Фортран 90/95 поддерживает значительную часть методологии ООП, хотя и не обладает полным набором средств ООП.

Фортран 2003 содержит средства, обеспечивающие полную поддержку ООП: введены расширяемые типы для реализации наследования и средства реализации динамического полиморфизма — полиморфные объекты, т.е. объекты, тип которых варьируется во время выполнения программы, конструкция SELECT TYPE, процедуры, привязанные к типу, и др.

4.8. Средства поддержки параллельности. Средства параллельного программирования впервые были введены в стандарт языка в Фортране 90. Необходимость их введения была вызвана появлением в компьютерах аппаратных средств векторной обработки. Эти свойства аппаратуры, в свою очередь, формировались на основе требований решаемых задач. Для таких компьютеров разрабатывались различные средства, включая и расширения Фортрана, однако использование нестандартных средств снижало мобильность программ; это привело к необходимости унифицировать подобные средства в стандарте языка Фортран.

В Фортран 90 были введены средства явной спецификации векторных операций. Это следующие черты. Возможность работы с массивами и секциями массивов как с целыми объектами на поэлементной основе, условное присваивание массиву под управлением логической маски (оператор WHERE и конструкция WHERE–END WHERE), встроенные поэлементные процедуры, аргументами которых могут быть как скаляры, так и массивы — все эти черты фактически специфицируют параллелизм действий над элементами массива или нескольких массивов. В язык добавлен большой набор встроенных процедур для работы с массивами; среди них часто используемые математические функции, которые входят в состав самого языка и реализуются компилятором, что позволяет эффективно использовать особенности архитектуры, включая и параллельное выполнение.

В Фортране 95 средства поддержки параллельности получили дальнейшее развитие. В язык дополнительно введены оператор и конструкция FORALL, которые позволяют специфицировать параллельный цикл. Введена спецификация PURE для функции без побочного эффекта; вызов такой функции можно использовать в тех случаях, где возможна параллельная обработка. Разрешено использовать не только встроенные поэлементные процедуры (как в Фортране 90), но также и поэлементные процедуры, определяемые в программе. Поскольку результат выполнения поэлементных функций и подпрограмм не зависит от порядка выбора элементов массивов, они могут выполняться параллельно.

Помимо указанных выше средств поддержки параллельности, в современных стандартах Фортрана имеются и другие черты, косвенно влияющие на параллельность. Разработанные для многопроцессорных систем средства параллельного программирования (OpenMP, HPF, MPI) являются фактически расширениями современного Фортрана, они существенно используют новые возможности языка. Кроме того, современный Фортран (в отличие от предшественников) позволяет, как уже отмечалось, разрабатывать программы лучше структурированные, более наглядные; такие программы легче распараллелить. В то же время многие архаизмы могут серьезно затруднить или сделать невозможной распараллеливание программы. Таким образом, программа, написанная на современном Фортране и не использующая устарев-

шие черты, лучше поддается распараллеливанию, чем программа, написанная на "старом" Фортране.

Хотя Фортран 90/95/2003 содержит некоторые средства поддержки параллельности, стандарт языка Фортран пока не ориентирован на многопроцессорную обработку и для параллельного программирования используются специальные языки — расширения Фортрана. В настоящее время уже принято решение ввести непосредственно в Фортран средства, поддерживающие параллельную обработку для многопроцессорной архитектуры (coarrays). Такие средства планируется ввести в следующий стандарт языка (Фортран 2008), проект которого опубликован и находится в стадии обсуждения.

5. Новые черты современного Фортрана. Новые черты Фортрана 77 (по сравнению с Фортраном 66) подробно описаны в [23]. Ниже перечислены наиболее значимые новые черты более поздних стандартов языка.

5.1. Новые черты языка Фортран 90. В данном подразделе перечислены некоторые новые (по сравнению с Фортраном 77) средства языка Фортран 90:

— свободный формат исходного текста;

— длинные имена;

— параметризованные встроенные типы;

— структурные типы данных (производные типы);

— новый вид объявления данных;

— средства работы с полными массивами и секциями массивов на поэлементной основе (явная спецификация векторных операций);

— механизмы динамического размещения объектов в памяти;

— указатели;

— современные управляющие конструкции;

— операции и присваивания, определяемые в программе;

— средства явного интерфейса — интерфейсные блоки;

— ключевые и необязательные аргументы процедур;

— спецификация аргументов по характеру использования;

— перегрузка операций и процедур;

— рекурсивные процедуры;

— внутренние процедуры;

— большой набор новых встроенных процедур;

— программные единицы-модули.

Кроме серьезных расширений, в Фортране 90 принята концепция эволюционного развития языка (см. раздел 6).

5.2. Новые черты языка Фортран 95. В Фортране 95 по сравнению с Фортраном 90 новшеств относительно немного, большая их часть связана с параллельностью. Ниже перечислены некоторые новые черты:

— оператор и конструкция FORALL (параллельный цикл);

— расширение возможностей конструкции WHERE;

— поэлементные процедуры, определяемые в программе;

— спецификация процедур без побочного эффекта (PURE-процедуры);

— инициализация указателей;

— две новые встроенные процедуры и дополнительные аргументы для некоторых встроенных процедур.

Помимо введения новых возможностей, устранены некоторые дефекты и нерегулярности и удалены некоторые устаревшие средства (см. раздел 6).

5.3. Новые черты языка Фортран 2003. Фортран 2003 содержит существенные нововведения. Ниже перечислены некоторые из наиболее значимых новых направлений:

— развитие средств объектно-ориентированного программирования (полный набор средств ООП);

— средства взаимодействия с языком Си, обеспечивающие как вызов Си-функций из Фортран-программы, так и вызов Фортран-процедур из Си-программы;

— параметризованные производные типы;

— новые средства ввода/вывода (асинхронный ввод/вывод, потоковый доступ к файлам, новые спецификаторы формата и др.);

— средства обработки исключительных ситуаций для операций с плавающей точкой;

— новые возможности, касающиеся динамически размещаемых массивов (для формальных аргументов, результатов функций, компонентов структуры);

— более полная интеграция с операционной системой (доступ к аргументам командной строки, к переменным окружения);

— встроенные модули и новые встроенные процедуры.

6. Устаревшие черты. После введения новых элементов некоторые более ранние конструкции языка Фортран стали излишними; такие конструкции отнесены к категории устаревших черт. Исключение из языка таких элементов может осложнить использование ранее разработанных программ на Фортране. В то же время, очевидно, что если в язык будут добавляться новые средства без удаления уже ненужных, устаревших элементов, язык станет очень громоздким с большим числом дублирующих элементов.

При разработке Фортрана 90 была принята следующая концепция: элементы языка, которые являются кандидатами на удаление, заранее объявляются устаревшими, не рекомендуемыми для использования.

В Фортране 90 утвержден список устаревших средств, но никакие средства не были удалены. В стандарте Фортран 95 расширен список устаревших средств и удалены следующие черты (некоторые из тех, которые в Фортране 90 были признаны устаревшими): параметры цикла вещественного типа и типа двойной точности; передача управления на оператор END IF извне данной конструкции; оператор паузы (PAUSE); оператор присваивания метки (ASSIGN) и оператор GO TO по предписанию; присваивание значения метки оператора FORMAT в операторе ASSIGN; описатель редактирования nH.

В Фортране 2003 удалено средство управления кареткой с помощью зарезервированных символов. В Фортране 2008 к списку устаревших средств предполагается добавить оператор ENTRY.

Список устаревших и удаленных черт и рекомендации по их замене современными средствами можно найти, например, в [6].

7. Операторы Фортрана. В табл. 1 перечислены операторы в разных стандартах языка в алфавитном порядке по их начальным ключевым словам. Операторы, не начинающиеся с ключевых слов, помещены в конце таблицы.

Операторы, помеченные символом звездочка “⋆” в столбце 3 (Вид), являются выполняемыми, остальные операторы — невыполняемые; “Атр” означает, что данное ключевое слово может использоваться (начиная с Фортрана 90) и для указания атрибута в операторах объявления типа. В столбцах 4 – 8 символ “+” означает, что оператор присутствует в данном варианте языка, а символ “—” означает, что оператор отсутствует. “Устар” означает, что в соответствующем стандарте оператор признан устаревшим, а “Удален” — оператор удален.

Таблица 1

Операторы в разных стандартах языка Фортран

Оператор 1	Назначение 2	Вид 3	Ф66 4	Ф77 5	Ф90 6	Ф95 7	Ф03 8
ALLOCATABLE	Объявление размещаемых массивов	Атр	—	—	+	+	+
ALLOCATE	Размещение размещаемых массивов	⋆	—	—	+	+	+
ASSIGN	Оператор присваивания метки	⋆	+	+	+ Устар	— Удален	—
ASSOCIATE	Начальный оператор конструкции связывания	⋆	—	—	—	—	+
ASYNCHRONOUS	Объявление объектов асинхронного ввода/вывода	Атр	—	—	—	—	+
BACKSPACE	Возврат указателя файла на одну запись	⋆	+	+	+	+	+
BIND	Объявление объектов для связи с Си-переменными	Атр	—	—	—	—	+
BLOCK DATA	Заголовок программной единицы-блока данных		+	+	+	+	+

Продолжение таблицы 1

1	2	3	4	5	6	7	8
CALL	Вызов подпрограммы	⋆	+	+	+	+	+
CASE	Вариант конструкции выбора	⋆	—	—	+	+	+
CASE DEFAULT	Вариант конструкции выбора по умолчанию	⋆	—	—	+	+	+
CHARACTER	Объявление символьного типа		—	+	+	+	+
CLOSE	Закрытие файла	⋆	—	+	+	+	+
COMMON	Объявление общих блоков		+	+	+	+	+
COMPLEX	Объявление данных комплексного типа		+	+	+	+	+
CONTAINS	Оператор отделяет внутренние или модульные процедуры		—	—	+	+	+
CONTINUE	Оператор используется для того, чтобы поместить метку	⋆	+	+	+	+	+
CYCLE	Переход к следующей итерации цикла	⋆	—	—	+	+	+
DATA	Оператор инициализации данных		+	+	+	+	+
DEALLOCATE	Удаление размещенных объектов	⋆	—	—	+	+	+
DIMENSION	Оператор объявления массивов	Атр	+	+	+	+	+
DO	Заголовок цикла	⋆	+	+	+	+	+
DO ... WHILE	Заголовок цикла с условием	⋆	—	—	+	+	+
DOUBLE PRECISION	Объявление данных типа двойной точности		+	+	+	+	+
ELSE	Альтернатива в конструкции условного перехода	⋆	—	+	+	+	+
ELSE IF	Вложенный оператор условия в конструкции условного перехода	⋆	—	+	+	+	+
ELSEWHERE	Альтернатива в конструкции присваивания по маске	⋆	—	—	+	+	+
END END PROGRAM END FUNCTION END SUBROUTINE END MODULE END BLOCK DATA	Заключительный оператор программной единицы, внутренней или модульной процедуры	⋆	—	—	+	+	+
END ASSOCIATE	Заключительный оператор конструкции связывания	⋆	—	—	—	—	+
END DO	Конец цикла	⋆	—	—	+	+	+
END ENUM	Конец описания перечислимого типа		—	—	—	—	+
ENDFILE	Запись в файл признака конца файла	⋆	+	+	+	+	+
END FORALL	Конец конструкции FORALL	⋆	—	—	—	+	+
END IF	Заключительный оператор конструкции условного перехода	⋆	—	+	+	+	+
END INTERFACE	Конец интерфейсного блока		—	—	+	+	+
END SELECT	Конец конструкции выбора	⋆	—	—	+	+	+
END TYPE	Конец описания производного типа		—	—	+	+	+

Продолжение таблицы 1

1	2	3	4	5	6	7	8
END WHERE	Конец конструкции присваивания по маске	⋆	—	—	+	+	+
ENTRY	Дополнительный вход в процедуру		—	+	+	+	+
ENUM	Начало конструкции описаний перечислимых типов		—	—	—	—	+
ENUMERATOR	Оператор описания перечислимого типа		—	—	—	—	+
EQUIVALENCE	Оператор эквивалентности		+	+	+	+	+
EXIT	Выход из цикла	⋆	—	—	+	+	+
EXTERNAL	Объявление имен внешних процедур	Атр	+	+	+	+	+
FINAL	Объявление финальных процедур		—	—	—	—	+
FLUSH	Оператор передачи данных	⋆	—	—	—	—	+
FORALL	Заголовок оператора или конструкции параллельный цикл	⋆	—	—	—	+	+
FORMAT	Объявление формата		+	+	+	+	+
FUNCTION	Заголовок внешней, внутренней или модульной функции		+	+	+	+	+
GENERIC	Объявление процедур с родовым именем		—	—	—	—	+
GOTO	Оператор безусловного перехода	⋆	+	+	+	+	+
GOTO по предписанию	Переход по предписанию	⋆	+	+	+ Устар	— Удален	—
GO TO со списком меток	Переход вычисляемый	⋆	+	+	+	+ Устар	+ Устар
IF	Оператор условного перехода (логический IF)	⋆	+	+	+	+	+
IF арифметический	Условный переход в зависимости от значения числового выражения	⋆	+	+	+ Устар	+ Устар	+ Устар
IF ... THEN	Начальный оператор конструкции условного перехода	⋆	—	+	+	+	+
IMPLICIT	Неявное объявление типа		—	+	+	+	+
IMPLICIT NONE	Отмена приписывания типа по умолчанию		—	—	+	+	+
IMPORT	Объявление импортируемых объектов		—	—	—	—	+
INCLUDE	Директива включения текста		—	—	+	+	+
INQUIRE	Запрос характеристик файла	⋆	—	+	+	+	+

Продолжение таблицы 1

1	2	3	4	5	6	7	8
INTEGER	Объявление данных целого типа		+	+	+	+	+
INTENT	Объявление назначения формальных аргументов	Атр	—	—	+	+	+
INTERFACE	Заголовок интерфейсного блока		—	—	+	+	+
INTRINSIC	Объявление встроенных процедур	Атр	—	+	+	+	+
LOGICAL	Объявление данных логического типа		+	+	+	+	+
MODULE	Заголовок программной единицы — модуля		—	—	+	+	+
MODULE PROCEDURE	Объявление имен модульных процедур		—	—	+	+	+
NAMELIST	Объявление именованного списка		—	—	+	+	+
NULLIFY	Разрывает связь указателя с объектом	⋆	—	—	+	+	+
OPEN	Открытие файла	⋆	—	+	+	+	+
OPTIONAL	Объявление необязательных аргументов	Атр	—	—	+	+	+
PARAMETER	Объявление именованных констант	Атр	—	+	+	+	+
PAUSE	Оператор паузы	⋆	+	+	+ Устар	— Удален	—
POINTER	Объявление указателей	Атр	—	—	+	+	+
PRINT	Оператор вывода на стандартное устройство	⋆	—	+	+	+	+
PRIVATE	Объявление локальных объектов	Атр	—	—	+	+	+
PROCEDURE	Оператор объявления процедур		—	—	—	—	+
PROGRAM	Заголовок главной программной единицы		—	+	+	+	+
PROTECTED	Объявление объектов с ограниченным доступом	Атр	—	—	—	—	+
PUBLIC	Объявление доступных объектов	Атр	—	—	+	+	+
READ	Оператор ввода (чтения)	⋆	+	+	+	+	+
REAL	Оператор объявления данных вещественного типа		+	+	+	+	+
RETURN	Возврат из процедуры	⋆	+	+	+	+	+
RETURN e	Альтернативный возврат из подпрограммы	⋆	—	+	+	+ Устар	+ Устар

Окончание таблицы 1

1	2	3	4	5	6	7	8
REWIND	Установка указателя файла в начало	$\star$	+	+	+	+	+
SAVE	Объявление сохраняемых объектов	Атр	—	+	+	+	+
SELECT CASE	Начальный оператор конструкции выбора блока операторов	$\star$	—	—	+	+	+
SELECT TYPE	Начальный оператор конструкции выбора блока в зависимости от динамического типа	$\star$	—	—	—	—	+
SEQUENCE	Объявление последовательной структуры		—	—	+	+	+
STOP	Оператор останова	$\star$	+	+	+	+	+
SUBROUTINE	Заголовок подпрограммы		+	+	+	+	+
TARGET	Объявление адресатов указателя	Атр	—	—	+	+	+
TYPE	Начальный оператор описания производного типа		—	—	+	+	+
TYPE (*mun*)	Объявление данных производного типа (указанного в скобках)		—	—	+	+	+
VALUE	Спецификация способа связи аргументов процедуры	Атр	—	—	—	—	+
VOLATILE	Объявление данных, которые могут изменяться средствами, не специфицированными в программе	Атр	—	—	—	—	+
USE	Объявление используемого модуля и глобальных объектов		—	—	+	+	+
WAIT	Оператор ожидания при асинхронной передаче данных	$\star$	—	—	—	—	+
WHERE	Оператор присваивания по маске или начальный оператор конструкции	$\star$	—	—	+	+	+
WRITE	Оператор вывода (записи)	$\star$	+	+	+	+	+
Оператор объявления операторной функции	Оператор объявления операторной функции		+	+	+	+ Устар	+ Устар
Оператор присваивания	Оператор присваивания	$\star$	+	+	+	+	+
Оператор присваивания указателю	Оператор присваивания указателю	$\star$	—	—	+	+	+

8. Встроенные процедуры. В табл. 2 приводится (в алфавитном порядке) список встроенных процедур в разных стандартах языка Фортран и краткое описание назначения каждой процедуры.

Квадратные скобки используются для необязательных аргументов (параметров).

В третьем столбце используются следующие обозначения: Ф Э — поэлементная функция, Ф С — справочная функция, Ф П — функция преобразования, ПП — подпрограмма.

В столбцах 4–7 символ “+” означает, что процедура имеется в данном стандарте, а символ “—” означает, что процедура отсутствует.

Таблица 2

Встроенные процедуры в разных стандартах языка Фортран

Процедура 1	Назначение 2	Вид 3	Ф77 4	Ф90 5	Ф95 6	Ф03 7
ABS (A)	Абсолютное значение	Ф Э	+	+	+	+
ACHAR (I [, KIND**])	Символ в позиции I кода ASCII	Ф Э	—	+	+	+
ACOS (X)	Арккосинус	Ф Э	+	+	+	+
ADJUSTL (STRING)	Смещение влево	Ф Э	—	+	+	+
ADJUSTR (STRING)	Смещение вправо	Ф Э	—	+	+	+
AIMAG (Z)	Мнимая часть комплексного числа	Ф Э	+	+	+	+
AINT (A [, KIND])	Усечение до целого числа	Ф Э	+	+	+	+
ALL (MASK [, DIM])	“Истина”, если все элементы “истина”, иначе — “ложь”	Ф П	—	+	+	+
ALLOCATED (ARRAY)	“Истина”, если массив размещен	Ф С	—	+	+	+
ALLOCATED (SCALAR)	“Истина”, если SCALAR размещен	Ф С	—	—	—	+
ANINT (A [, KIND])	Ближайшее к вещественному аргументу целое число (в форме веществ. числа)	Ф Э	+	+	+	+
ANY (MASK [, DIM])	“Истина”, если хотя бы один элемент “истина”, иначе — “ложь”	Ф П	—	+	+	+
ASIN (X)	Арксинус	Ф Э	+	+	+	+
ASSOCIATED (POINTER [, TARGET])	Статус связанности указателя	Ф С	—	+	+	+
ATAN (X)	Арктангенс	Ф Э	+	+	+	+
ATAN2 (Y, X)	Арктангенс комплексного аргумента	Ф Э	+	+	+	+
BIT_SIZE (I)	Число битов в модели	Ф С	—	+	+	+
BTEST (I, POS)	Проверка бита	Ф Э	—	+	+	+
CEILING (A [, KIND*])	Наименьшее целое, большее или равное аргументу	Ф Э	—	+	+	+
CHAR (I [, KIND])	Символ в заданной позиции I в кодировке процессора	Ф Э	+	+	+	+
CMPLX (X, Y [, KIND])	Преобразование в комплексный тип	Ф Э	+	+	+	+
COMMAND_ARGUMENT_COUNT ( )	Число элементов в командной строке	Ф С	—	—	—	+
CONJG (Z)	Сопряженное комплексное число	Ф Э	+	+	+	+
COS (X)	Косинус	Ф Э	+	+	+	+

Продолжение таблицы 2

1	2	3	4	5	6	7
COSH (X)	Гиперболический косинус	Ф Э	+	+	+	+
COUNT (MASK [, DIM] [, KIND**])	Число элементов со значением “истина”	Ф П	—	+	+	+
CPU_TIME (TIME)	Процессорное время	ПП	—	—	+	+
CSHIFT (ARRAY, SHIFT [, DIM])	Циклический сдвиг	Ф П	—	+	+	+
DATE_AND_TIME ([DATE] [, TIME] [, ZONE] [, VALUES])	Чтение даты и времени	ПП	—	+	+	+
DBLE (A)	Преобразование к вещественному типу двойной точности	Ф Э	+	+	+	+
DIGITS (X)	Число значащих цифр в модельном представлении аргумента	Ф С	—	+	+	+
DIM (X, Y)	Положительная разность max (X-Y, 0)	Ф Э	+	+	+	+
DOT_PRODUCT (VECTOR_A, VECTOR_B)	Скалярное произведение векторов	Ф П	—	+	+	+
DPROD (X, Y)	Произведение двух вещественных, результат — двойной точности	Ф Э	+	+	+	+
EOSHIFT (ARRAY, SHIFT [, BOUNDARY] [, DIM])	Нециклический сдвиг	Ф П	—	+	+	+
EPSILON (X)	Почти не отличающееся от единицы число для модели X	Ф С	—	+	+	+
EXP (X)	Экспонента	Ф Э	+	+	+	+
EXPONENT (X)	Степенная часть модельного представления аргумента (порядок)	Ф Э	—	+	+	+
EXTENDS_TYPE_OF (A, MOLD)	Проверяет, является ли динамический тип A расширением динамического типа MOLD	Ф С	—	—	—	+
FLOOR (A [, KIND*])	Наибольшее целое, меньшее или равное аргументу	Ф Э	—	+	+	+
FRACTION (X)	Дробная часть модельного представления аргумента	Ф Э	—	+	+	+
GET_COMMAND ([COMMAND] [, LENGTH] [, STATUS])	Возвращает командную строку	ПП	—	—	—	+
GET_COMMAND_ARGUMENT (NUMBER [, VALUE] [, LENGTH] [, STATUS])	Возвращает один аргумент командной строки	ПП	—	—	—	+
GET_ENVIRONMENT_VARIABLE (NAME [, VALUE] [, LENGTH] [, STATUS] [, TRIM_NAME])	Выдает значение переменной окружения	ПП	—	—	—	+
HUGE (X)	Наибольшее число в модели представления аргумента	Ф С	—	+	+	+

Продолжение таблицы 2

1	2	3	4	5	6	7
IACHAR (C [, KIND**])	Позиция символа в кодировке ASCII	Ф Э	—	+	+	+
IAND (I, J)	Логическое AND (поразрядное)	Ф Э	—	+	+	+
IBCLR (I, POS)	Обнуление бита POS	Ф Э	—	+	+	+
IBITS (I, POS, LEN)	Выделение последовательности битов	Ф Э	—	+	+	+
IBSET (I, POS)	Установка бита POS равным единице	Ф Э	—	+	+	+
ICHAR (C [, KIND**])	Номер позиции символа C в кодировке процессора	Ф Э	+	+	+	+
IEOR (I, J)	Исключающее OR для битов	Ф Э	—	+	+	+
INDEX (STRING, SUBSTRING [, BACK] [, KIND**])	Начальная позиция подстроки в строке символов	Ф Э	—	+	+	+
INT(A [, KIND])	Преобразование в целый тип	Ф Э	+	+	+	+
IOR (I, J)	Включающее OR для битов	Ф Э	—	+	+	+
ISHFT (I, SHIFT)	Логический сдвиг для битов	Ф Э	—	+	+	+
ISHFTC (I, SHIFT [, SIZE])	Циклический сдвиг набора битов	Ф Э	—	+	+	+
IS_IOSTAT_END (I)	Проверяет значение состояния “конец файла”	Ф Э	—	—	—	+
IS_IOSTAT_ERR (I)	Проверяет значение состояния “конец записи”	Ф Э	—	—	—	+
KIND (X)	Значение параметра разновидности типа	Ф С	—	+	+	+
LBOUND (ARRAY [, DIM] [, KIND**])	Нижние границы измерений массива	Ф С	—	+	+	+
LEN (STRING [, KIND**])	Длина символьного аргумента	Ф С	+	+	+	+
LEN_TRIM (STRING [, KIND**])	Возвращает длину строки за вычетом конечных пробелов	Ф Э	—	+	+	+
LGE (STRING_A, STRING_B)	Лексически больше или равно в ASCII последовательности	Ф Э	+	+	+	+
LGT (STRING_A, STRING_B)	Лексически больше в последовательности ASCII	Ф Э	+	+	+	+

Продолжение таблицы 2

1	2	3	4	5	6	7
LLE (STRING_A, STRING_B)	Лексически меньше или равно в ASCII последовательности	Ф Э	+	+	+	+
LLT (STRING_A, STRING_B)	Лексически меньше в последовательности ASCII	Ф Э	+	+	+	+
LOG (X)	Натуральный логарифм	Ф Э	+	+	+	+
LOG10 (X)	Десятичный логарифм	Ф Э	+	+	+	+
LOGICAL (L [, KIND])	Преобразование объектов логического типа с разными параметрами типа	Ф Э	+	+	+	+
MATMUL (MATRIX_A, MATRIX_B)	Умножение двух матриц, матрицы на вектор или вектора на матрицу	Ф П	—	+	+	+
MAX (A1, A2 [, A3,...])	Максимальное значение	Ф Э	+	+	+	+
MAXEXPONENT (X)	Максимальная экспонента (порядок) в модели представления X	Ф С	—	+	+	+
MAXLOC (ARRAY [, DIM*] [, MASK] [, KIND**])	Положение максимального элемента в массиве	Ф П	—	+	+	+
MAXVAL (ARRAY [, DIM] [, MASK])	Значение максимального элемента массива	Ф П	—	+	+	+
MERGE (TSOURCE, FSOURCE, MASK)	Слияние массивов путем выбора альтернативного значения в соответствии со значением логической маски	Ф Э	—	+	+	+
MIN (A1, A2 [, A3,...])	Минимальное значение	Ф Э	+	+	+	+
MINEXPONENT (X)	Минимальная экспонента в модели представления X	Ф С	—	+	+	+
MINLOC (ARRAY [, DIM*] [, MASK] [, KIND**])	Положение минимального элемента в массиве	Ф П	—	+	+	+
MINVAL (ARRAY [, DIM] [, MASK])	Значение минимального элемента массива	Ф П	—	+	+	+
MOD (A, P)	Функция остатка, т.е. A-INT(A/P)$\star$P	Ф Э	+	+	+	+
MODULO (A, P)	Функция “по модулю”	Ф Э		+	+	+
MOVE_ALLOC (FROM, TO)	Передача размещения объекта	ПП	—	—	—	+
MVBITS (FROM, FROMPOS, LEN, TO, TOPOS)	Копирование битов	ПП	—	+	+	+
NEAREST (X, S)	Ближайшее, отличное от аргумента процессорное число в заданном направлении (направление определяется знаком S)	Ф Э	—	+	+	+
NEW_LINE (A)	Символ новой строки	Ф С	—	—	—	+

Продолжение таблицы 2

1	2	3	4	5	6	7
NINT (A [, KIND])	Целое, ближайшее к вещественному аргументу	Ф Э	+	+	+	+
NOT (I)	Логическое дополнение для битов	Ф Э	—	+	+	+
NULL ([MOLD])	Функция статуса связанности указателя	Ф П	—	—	+	+
PACK (ARRAY, MASK [, VECTOR])	Упаковка массива в вектор под управлением логической маски	Ф П	—	+	+	+
PRECISION (X)	Десятичная точность в модельном представлении X	Ф С	—	+	+	+
PRESENT (X)	“Истина”, если необязательный аргумент присутствует, иначе —“ложь”	Ф С	—	+	+	+
PRODUCT (ARRAY [, DIM] [, MASK])	Произведение элементов массива	Ф П	—	+	+	+
RADIX (X)	Основание системы счисления модели представления X	Ф С	—	+	+	+
RANDOM_NUMBER (HARVEST)	Инициализация генератора псевдослучайных чисел	ПП	—	+	+	+
RANDOM_SEED ([SIZE] [, PUT] [, GET])	Генератор псевдослучайных чисел	ПП	—	+	+	+
RANGE (X)	Значение диапазона десятичной экспоненты	Ф С	—	+	+	+
REAL (A [, KIND])	Преобразование в вещественный тип	Ф Э	+	+	+	+
REPEAT (STRING, NCOPIES)	Конкатенация копий аргумента STRING	Ф П	—	+	+	+
RESHAPE (SOURCE, SHAPE [, PAD] [, ORDER])	Изменение конфигурации массива	Ф П	—	+	+	+
RRSPACING (X)	Обратная величина относительного расстояния между числами модели в области X	Ф Э	—	+	+	+
SAME_TYPE_AS (A, B)	Сравнение динамических типов	Ф С	—	—	—	+
SCALE (X, I)	Умножение вещественного числа на основание системы счисления, возведенное в целую степень, определяемую аргументом I ($X \star b^I$)	Ф Э	—	+	+	+
SCAN (STRING, SET [, BACK] [, KIND**])	Номер позиции символа из набора SET в аргументе STRING	Ф Э	—	+	+	+
SELECTED_CHAR_KIND (NAME)	Значение параметра типа для символьного аргумента	Ф П	—	—	—	+

Продолжение таблицы 2

1	2	3	4	5	6	7
SELECTED_INT_KIND (R)	Значение параметра разновидности типа для целых в заданном диапазоне	Ф П	—	+	+	+
SELECTED_REAL_KIND ([P] [, R])	Значение параметра разновидности типа для вещественных с заданными точностью и диапазоном	Ф П	—	+	+	+
SET_EXPONENT (X, I)	Установить экспонентную часть числа модельного представления аргумента	Ф Э	—	+	+	+
SHAPE (SOURCE [, KIND**])	Конфигурация массива или скаляра	Ф С	—	+	+	+
SIGN (A, B)	Передача знака	Ф Э	—	+	+	+
SIN (X)	Синус	Ф Э	+	+	+	+
SINH (X)	Гиперболический синус	Ф Э	+	+	+	+
SIZE (ARRAY [, DIM] [, KIND**])	Общее число элементов массива	Ф С	—	+	+	+
SPACING (X)	Абсолютное расстояние между модельными числами области X	Ф Э	—	+	+	+
SPREAD (SOURCE, DIM, NCOPIES)	Расширение массива путем добавления копий указанного измерения	Ф П	—	+	+	+
SQRT (X)	Корень квадратный	Ф Э	+	+	+	+
SUM (ARRAY [, DIM] [, MASK])	Сумма элементов массива	Ф П	—	+	+	+
SYSTEM_CLOCK ([COUNT] [, COUNT_RATE] [, COUNT_MAX])	Время системных часов	ПП	—	+	+	+
TAN (X)	Тангенс	Ф Э	+	+	+	+
TANH (X)	Гиперболический тангенс	Ф Э	+	+	+	+
TINY (X)	Наименьшее положительное число в модельном представлении аргумента	Ф С	—	+	+	+
TRANSFER (SOURCE, MOLD [, SIZE])	Передача типа	Ф П	—	+	+	+
TRANSPOSE (MATRIX)	Транспонирование матрицы	Ф П	—	+	+	+
TRIM (STRING)	Удаление конечных пробелов	Ф П	—	+	+	+

Окончание таблицы 2

1	2	3	4	5	6	7
UBOUND (ARRAY [, DIM] [, KIND**])	Верхние границы измерений массива	Ф С	—	+	+	+
UNPACK (VECTOR, MASK, FIELD)	Распаковка вектора в массив под управлением логической маски	Ф П	—	+	+	+
VERIFY (STRING, SET [, BACK] [, KIND**])	Положение символа из STRING, которого нет в SET	Ф Э	—	+	+	+

Примечание. В табл. 2 символ "*" после имени аргумента означает, что в Фортране 90 отсутствует помеченный аргумент (аргумент добавлен в Фортране 95), двойной символ "**" после аргумента означает, что аргумент добавлен в Фортране 2003. Аргумент KIND в Фортране 77 отсутствует.

9. О проекте языка Фортран 2008. Развитие языка Фортран продолжается. Подготовлен рабочий вариант проекта будущего стандарта языка — Фортран 2008 [24]. Ниже перечислены некоторые новые черты этого проекта:

— поддержка параллельного программирования (coarrays);

— максимальная размерность (ранг) массива увеличивается с 7 до 15;

— конструкция BLOCK–END BLOCK, которая позволяет помещать объявления в блоке среди выполняемых операторов;

— конструкция DO CONCURENT, которая позволяет, чтобы итерации цикла выполнялись в произвольном порядке, или потенциально параллельно;

— некоторые расширения, касающиеся ввода-вывода;

— дополнительные встроенные математические функции для вычисления гамма-функции, бесселевых функций, функции ошибки и др;

— новые встроенные процедуры (для поддержки параллельности, битового типа и др.) и добавленные аргументы в некоторых встроенных процедурах предыдущих стандартов;

— функции COMPILER_VERSION и COMPILER_OPTIONS, обеспечивающие информацию о фазе трансляции при выполнении программ;

— средства улучшения регулярности языка (например, выбор действительной и мнимой частей комплексной переменной аналогично выбору компонент производного типа и др.).

СПИСОК ЛИТЕРАТУРЫ

1. ANSI X3.9-1966. USA Standard FORTRAN.
2. ANSI X3.9-1978. American National Standard — Programming Language FORTRAN (ISO 1539-1980).
3. ISO/IEC 1539: 1991(E). Information technology — Programming languages — Fortran.
4. Фортран 90. Международный стандарт / Перевод с англ. С.Г. Дробышевич, редактор перевода А.М. Горелик. М.: Финансы и статистика, 1998.
5. ISO/IEC 1539-1: 1997. Information technology — Programming languages — Fortran. Part1: Base Language.
6. *Горелик А.М.* Программирование на современном Фортране. М.: Финансы и статистика, 2006.
7. *Горелик А.М.* Современный Фортран для компьютеров традиционной архитектуры и для параллельных вычислительных систем // Вычислительные методы и программирование. 2004. **5**, № 1. 137–149.
8. ISO/IEC 1539-1: 2004. Information technology — Programming languages — Fortran. Part1: Base Language.
9. *Балацкова-Подольскова С.И., Булко И.М., Цагельский В.И.* Фортран ЭВМ "Минск-32". М.: Статистика, 1975.
10. *Салтыков А.И., Макаренко Г.И.* Программирование на языке Фортран. М.: Наука, 1976.
11. *Горелик А.М.* Входной язык шагового транслятора с Фортрана // Журн. вычисл. матем. и матем. физики. 1975. **15**, № 2. 457–467.
12. *Горелик А.М., Хухлаев Е.В.* Реализация шагового транслятора с Фортрана // Журн. вычисл. матем. и матем. физики. 1975. **15**, № 3. 728–736.
13. Расширенный ФОРТРАН-ФОРЕКС. Руководство для пользователя / Ю.М. Баяковский, Н.Н. Вьюкова, В.А. Галатенко и др. М.: ИПМ им. М.В. Келдыша АН СССР, 1983.
14. *Галатенко В.А., Ходулев А.В.* Реализация транслятора Форекс для ЦП АС-6 // Программирование. 1981. № 5. 50–58.
15. *Брич З.С., Капилевич Д.В., Котик С.Ю., Цагельский В.И.* Фортран ЕС ЭВМ. М.: Финансы и статистика, 1978.

16. Фортран 77 ЕС ЭВМ / Брич З.С., Гулецкая О.Н., Капилевич Д.В. и др. М.: Финансы и статистика, 1989.
17. Малые ЭВМ и их применение / Под ред. Б. Н. Наумова. М.: Статистика, 1980.
18. *Голосов И.С.* Фортран для вычислительных комплексов "Эльбрус" // Проблемы создания супер-ЭВМ, суперсистем и эффективность их применения. Минск, 1987. 132–134.
19. *Камынин С.С., Любимский Э.З.* Алгоритмический машинно-независимый язык АЛМО // Алгоритмы и алгоритмические языки. Вып. 1. М.: ВЦ АН СССР, 1968.
20. *Баяковский Ю.М., Галактионов В.А., Михайлова Т.М.* Графор. Графическое расширение Фортрана. М.: Наука, 1985.
21. *Горелик А.М.* Анализ некоторых способов оценки качества трансляторов // Программирование. 1981. № 1. 79–82.
22. *Горелик А.М., Ушкова В.Л., Шура-Бура М.Р.* Мобильность программ на Фортране. М.: Финансы и статистика, 1984.
23. *Горелик А.М., Ушкова В.Л.* Фортран сегодня и завтра. М.: Наука, 1990.
24. http://www.nag.co.uk/sc22wg5

Поступила в редакцию
04.04.2008