Сравнение стандартов языка Фортран

А.М. Горелик

Аннотация. В статье рассматривается эволюция основных средств языка в стандартах Фортрана (от Фортрана 66 до Фортрана 2008). Перечисляются новшества, введенные в каждый последующий стандарт языка. Приводятся таблицы отличий операторов, типов данных и встроенных процедур в различных стандартах.

Ключевые слова: Фортран, стандарты, языки программирования.

Введение

Язык программирования Фортран занимает лидирующие позиции при решении научно-технических и инженерных задач, требующих большого объема вычислений.

Фортран постоянно развивается и совершенствуется в соответствии с развитием вычислительной техники, языков и технологий программирования.

Язык Фортран шесть раз подвергался стандартизации на международном уровне (Фортран 66, Фортран 77, Фортран 90, Фортран 95, Фортран 2003 и Фортран 2008 [1–10]). При этом практически сохраняется преемственность с предыдущими стандартами языка, что позволяет использовать ранее созданные библиотеки и прикладные программы.

В статье перечисляются новшества, введенные в каждый последующий стандарт, и приводятся таблицы отличий основных элементов языка в различных стандартах (от Фортрана 66 до Фортрана 2008).

Более подробно исторический обзор развития языка Фортран (до принятия стандарта Фортрана 2008) приведен в работе [9].

1. Новые черты языка Фортран 90

В данном разделе перечислены основные новые (по сравнению с его предшественником 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, включая поэлементные.
- Программные единицы-модули.
- Принята концепция эволюционного развития языка.

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.

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 наоборот [11].
- 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.

4. Новые черты языка Фортран 2008

Ниже перечислены основные новые черты Фортрана 2008.

- Средства поддержки параллельных вычислений (coarrays, подр. [12]).
- Максимальный ранг (размерность) массива увеличен с 7 до 15.
- Конструкция BLOCK – END BLOCK, которая позволяет помещать объявления в блоке среди выполняемых операторов.
- Заголовок цикла (DO CONCURRENT), в котором нет зависимостей по данным между итерациями; это позволяет выполнять итерации цикла в произвольном порядке, или потенциально параллельно.
- Оператор EXIT допускает выход не только из цикла, но и из любой управляющей конструкции.
- Оператор CONTIGUOUS, который специфицирует, что указанный массив занимает смежный блок памяти.
- Оператор ERROR STOP для ошибочного завершения программы.
- Некоторые расширения, касающиеся ввода-вывода.
- Дополнительные встроенные математические функции для вычисления Гамма-функции, Бесселевых функций, функций ошибки и др.
- Новые встроенные процедуры и добавленные аргументы в некоторых встроенных процедурах предыдущих стандартов (для поддержки параллельности, битовые операции, новые функции редукции массивов и др.).
- Средства улучшения регулярности языка (например, выбор действительной и мнимой частей комплексной переменной аналогично выбору компонент производного типа).
- Подмодули.
- Оператор ENTRY отнесен к категории устаревших черт языка.

5. Типы данных в стандартах Фортрана

В приведенной ниже таблице перечислены типы данных в разных стандартах языка Фортран. Символ “+” означает, что тип присутствует в данном варианте языка, символ “–“, если отсутствует.

Тип	F66	F77	F90	F95	F03	F08
Целый	+	+	+	+	+	+
Вещественный	+	+	+	+	+	+
Вещественный двойной точности	+	+	+	+	+	+
Логический	+	+	+	+	+	+
Комплексный	+	+	+	+	+	+
Символьный	–	+	+	+	+	+
Параметризованный встроенный	–	–	+	+	+	+
Производный	–	–	+	+	+	+
Параметризованный производный	–	–	–	–	+	+
Динамический	–	–	–	–	+	+

6. Операторы Фортрана

В приведенной ниже таблице перечислены операторы в разных стандартах языка Фортран в алфавитном порядке по их начальным ключевым словам. Операторы, не начинающиеся с ключевых слов, помещены в конце таблицы.

Квадратные скобки используются для необязательных элементов.

В столбцах 3–8 символ “+” означает, что оператор присутствует в данном варианте языка, символ “–“, если оператор отсутствует.

“Obs” – (obsolescent) означает, что в соответствующем стандарте оператор признан устаревшим, а “Del” (deleted) – оператор удален.

Оператор	Назначение	F66	F77	F90	F95	F03	F08
ALLOCATABLE	Объявление размещаемых массивов	–	–	+	+	+	+
ALLOCATE	Размещение размещаемых массивов или комассивов	–	–	+	+	+	+
ASSIGN	Оператор присваивания метки	+	+	+ Obs	– Del	–	–
ASSOCIATE	Начальный оператор конструкции связывания	–	–	–	–	+	+
ASYNCHRONOUS	Объявление объектов асинхронного ввода/вывода	–	–	–	–	+	+
BACKSPACE	Возврат указателя файла на одну запись	+	+	+	+	+	+
BIND	Объявление объектов для связи с Си-переменными	–	–	–	–	+	+
BLOCK	Начальный оператор выполняемой конструкции, содержащей объявления	–	–	–	–	–	+
BLOCK DATA	Заголовок программной единицы-блока данных	+	+	+	+	+	+
CALL	Вызов подпрограммы	+	+	+	+	+	+
CASE	Вариант конструкции выбора	–	–	+	+	+	+

CASE DEFAULT	Вариант конструкции выбора по умолчанию	–	–	+	+	+	+
CHARACTER	Объявление символьного типа	–	+	+	+	+	+
CLASS	Объявление полиморфных объектов	–	–	–	–	+	+
CLOSE	Закрытие файла	–	+	+	+	+	+
CODIMENSION	Объявление комассивов	–	–	–	–	–	+
COMMON	Объявление общих блоков	+	+	+	+	+	+
COMPLEX	Объявление данных комплексного типа	+	+	+	+	+	+
CONTAINS	Оператор отделяет внутренние или модульные процедуры	–	–	+	+	+	+
CONTIGUOUS	Специфицирует, что указанный массив занимает смежный блок памяти	–	–	–	–	–	+
CONTINUE	Оператор используется для того, чтобы поместить метку	+	+	+	+	+	+
CRITICAL	Начало конструкции критической секции	–	–	–	–	–	+
CYCLE	Переход к следующей итерации цикла	–	–	+	+	+	+
DATA	Оператор инициализации данных	+	+	+	+	+	+
DEALLOCATE	Удаление размещенных объектов	–	–	+	+	+	+
DIMENSION	Оператор объявления массива	+	+	+	+	+	+
DO	Заголовок цикла	+	+	+	+	+	+
DO CONCURRENT	Заголовок цикла, в котором нет зависимостей по данным между итерациями	–	–	–	–	–	+
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	Заключительный оператор программной единицы	+	+	+	+	+	+
END ASSOCIATE	Заключительный оператор конструкции связывания	–	–	–	–	+	+
END BLOCK	Конечный оператор выполняемой конструкции, содержащей объявления	–	–	–	–	–	+
END [BLOCK DATA]	Конечный оператор блока данных	–	–	+	+	+	+
END CRITICAL	Конечный оператор критической секции	–	–	–	–	–	+
END DO	Конец цикла	–	–	+	+	+	+
END ENUM	Конец описания перечислимого типа	–	–	–	–	+	+
ENDFILE	Запись в файл признака конца файла	+	+	+	+	+	+
END FORALL	Конец конструкции FORALL	–	–	–	+	+	+
END [FUNCTION]	Конечный оператор внешней, внутренней или модульной функции	+	+	+	+	+	+
END IF	Заключительный оператор конструкции условного перехода	–	+	+	+	+	+
END INTERFACE	Конец интерфейсного блока	–	–	+	+	+	+
END [MODULE]	Конечный оператор программной единицы-модуля	–	–	+	+	+	+
END [PROGRAM]	Конечный оператор главной программной единицы	–	–	+	+	+	+
END SELECT	Конец конструкции выбора	–	–	+	+	+	+
END SUBMODULE	Конечный оператор подмодуля	–	–	–	–	–	+
END [SUBROUTINE]	Конечный оператор подпрограммы	–	–	+	+	+	+

END TYPE	Конец описания производного типа	–	–	+	+	+	+
END WHERE	Конец конструкции присваивания по маске	–	–	+	+	+	+
ENTRY	Дополнительный вход в процедуру	–	+	+	+	+	+ Obs
ENUM	Начало конструкции описаний перечислимых типов	–	–	–	–	+	+
ENUMERATOR	Оператор описания перечислимого типа	–	–	–	–	+	+
EQUIVALENCE	Оператор эквивалентности	+	+	+	+	+	+
ERROR STOP	Ошибочное завершение выполнения	–	–	–	–	–	+
EXIT	Выход из цикла	–	–	+	+	+	+
EXIT	Выход из любой управляющей конструкции	–	–	–	–	–	+
EXTERNAL	Объявление имен внешних процедур	+	+	+	+	+	+
FINAL	Объявление финальных процедур	–	–	–	–	+	+
FLUSH	Оператор передачи данных	–	–	–	–	+	+
FORALL	Заголовок оператора или конструкции параллельный цикл	–	–	–	+	+	+
FORMAT	Объявление формата	+	+	+	+	+	+
[*prefix*] FUNCTION	Заголовок внешней, внутренней или модульной функции	+	+	+	+	+	+
GENERIC	Объявление процедур с родовым именем	–	–	–	–	+	+
GOTO	Оператор безусловного перехода	+	+	+	+	+	+
GOTO assigned	Переход по предписанию	+	+	+ Obs	– Del	–	–
GO TO computed	Переход вычисляемый	+	+	+	+ Obs	+ Obs	+ Obs

IF	Оператор условного перехода (логический IF)	+	+	+	+	+	+
IF arithmetic	Условный переход в зависимости от значения числового выражения	+	+	+ Obs	+ Obs	+ Obs	+ Obs
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	–	+	+	+	+	+
INTEGER	Объявление данных целого типа	+	+	+	+	+	+
INTENT	Объявление назначения формальных аргументов	–	–	+	+	+	+
INTERFACE	Заголовок интерфейсного блока	–	–	+	+	+	+
INTRINSIC	Объявление встроенных процедур	–	+	+	+	+	+
LOCK	Оператор блокировки	–	–	–	–	–	+
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	–	–	+	+	+	+
PARAMETER	Объявление именованных констант	–	+	+	+	+	+
PAUSE	Оператор паузы	+	+	+ Obs	– Del	–	–
POINTER	Объявление указателей	–	–	+	+	+	+
PRINT	Оператор вывода на стандартное устройство	–	+	+	+	+	+
PRIVATE	Объявление локальных объектов	–	–	+	+	+	+
PROCEDURE	Оператор объявления процедур	–	–	–	–	+	+
PROGRAM	Заголовок главной программной единицы	–	+	+	+	+	+
PUBLIC	Объявление общих объектов	–	–	+	+	+	+
READ	Оператор ввода (чтения)	+	+	+	+	+	+
REAL	Оператор объявления данных вещественного типа	+	+	+	+	+	+
RETURN	Возврат из процедуры	+	+	+	+	+	+
RETURN *выражение*	Альтернативный возврат из подпрограммы	–	+	+ Obs	+ Obs	+ Obs	+ Obs
REWIND	Установка указателя файла в начало	+	+	+	+	+	+
SAVE	Объявление сохраняемых объектов	–	+	+	+	+	+
SELECT CASE	Начальный оператор конструкции выбора блока операторов	–	–	+	+	+	+
SELECT TYPE	Начальный оператор конструкции выбора динамического типа	–	–	–	–	+	+
SEQUENCE	Объявление последовательной структуры	–	–	+	+	+	+
STOP	Оператор останова	+	+	+	+	+	+

SUBMODULE	Заголовок подмодуля	–	–	–	–	–	+
[*prefix*] SUBROUTINE	Заголовок подпрограммы	+	+	+	+	+	+
SYNC ALL	Синхронизация всех экземпляров (images)	–	–	–	–	–	+
SYNC IMAGES	Синхронизация указанных экземпляров (images)	–	–	–	–	–	+
SYNC MEMORY	Завершает один сегмент и начинает другой	–	–	–	–	–	+
TARGET	Объявление адресатов указателя	–	–	+	+	+	+
TYPE	Начальный оператор описания производного типа	–	–	+	+	+	+
TYPE (*mun*)	Объявление данных производного типа	–	–	+	+	+	+
VALUE	Спецификация способа связи аргументов процедуры	–	–	–	–	+	+
VOLATILE	Объявление данных, которые могут изменяться средствами, не специфицированными в программе	–	–	–	–	+	+
UNLOCK	Оператор разблокировки	–	–	–	–	–	+
USE	Объявление используемого модуля и глобальных объектов	–	–	+	+	+	+
WAIT	Оператор ожидания при асинхронной передаче данных	–	–	–	–	+	+
WHERE	Оператор присваивания по маске или начальный оператор конструкции	–	–	+	+	+	+
WRITE	Оператор вывода	+	+	+	+	+	+
Оператор объявления операторной функции	Оператор объявления операторной функции	+	+	+	+ Obs	+ Obs	+ Obs

Оператор присваивания	Оператор присваивания	+	+	+	+	+	+
Оператор присваивания указателю	Оператор присваивания указателю	–	–	+	+	+	+

7. Встроенные процедуры

В таблице 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).

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, тройной символ “***”, если аргумент добавлен в Фортране 2008. Аргумент KIND в Фортране 77 отсутствует.

В третьем столбце первая буква указывает, что процедура является функцией или подпрограммой, вторая – указывает класс функции или подпрограммы. Используются следующие обозначения:

F – Функция (Function)
S – Подпрограмма (Subroutine)
E – Поэлементная процедура (Elemental)
I – Справочная функция (Inquire)
T – Функция преобразования (Transformational)
A – Атомарная подпрограмма (Atomic)

В столбцах 4–8 символ “+” означает, что процедура имеется в данном стандарте, символ “–”, если процедура отсутствует.

Процедура	Назначение	Вид	F77	F90	F95	F03	F08
ABS (A)	Абсолютное значение	FE	+	+	+	+	+
ACHAR (I [, KIND**])	Символ в позиции I кода ASCII	FE	–	+	+	+	+
ACOS (X)	Функция арккосинус	FE	+	+	+	+	+
ACOSH (X)	Гиперболический арккосинус	FE	–	–	–	–	+
ADJUSTL (STRING)	Смещение влево	FE	–	+	+	+	+
ADJUSTR (STRING)	Смещение вправо	FE	–	+	+	+	+
AIMAG (Z)	Мнимая часть комплексного числа	FE	+	+	+	+	+
AINT (A [, KIND])	Усечение до целого числа	FE	+	+	+	+	+
ALL (MASK [, DIM])	“Истина”, если все элементы “истина”, иначе - “ложь”	FT	–	+	+	+	+
ALLOCATED (ARRAY)	”Истина”, если массив размещен	FI	–	+	+	+	+

ALLOCATED (SCALAR)	”Истина”, если SCALAR размещен	F I	–	–	–	+	+
ANINT (A [, KIND])	Ближайшее к вещественному аргументу целое число (в форме вещественного)	F E	+	+	+	+	+
ANY (MASK [, DIM])	"Истина", если хотя бы один элемент "истина”, иначе - “ложь”	F T	–	+	+	+	+
ASIN (X)	Функция арксинус	F E	+	+	+	+	+
ASINH (X)	Гиперболический арксинус	F E	–	–	–	–	+
ASSOCIATED (POINTER [, TARGET])	Статус связанности указателя	F I	–	+	+	+	+
ATAN (X)	Функция арктангенс	F E	+	+	+	+	+
ATAN2 (Y, X)	Арктангенс комплексного аргумента	F E	+	+	+	+	+
ATANH (X)	Гиперболический арктангенс	F E	–	–	–	–	+
ATOMIC_DEFINE (ATOM, VALUE)	Определение значения переменной атомарно	S A	–	–	–	–	+
ATOMIC_REF(VALUE, ATOM)	Ссылка на переменную атомарно	S A	–	–	–	–	+
BESSEL_J0 (X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_J1 (X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_JN (N, X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_JN (N1, N2, X)	Функция Бесселя	F T	–	–	–	–	+
BESSEL_Y0 (X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_Y1 (X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_YN (N, X)	Функция Бесселя	F E	–	–	–	–	+
BESSEL_YN (N1, N2, X)	Функция Бесселя	F T	–	–	–	–	+
BGE (J, I) BGT (J, I) BLE (J, I) BLT (J, I)	Функции поразрядного сравнения	F E	– – – –	– – – –	– – – –	– – – –	+ + + +
BIT_SIZE (I)	Число битов в модели	F I	–	+	+	+	+

BTEST (I, POS)	Проверка бита	F E	–	+	+	+	+
CEILING (A [, KIND*])	Наименьшее целое, большее или равное аргументу	F E	–	+	+	+	+
CHAR (I [, KIND])	Символ в позиции I в кодировке процессора	F E	+	+	+	+	+
CMPLX (X [, Y] [, KIND])	Преобразование в комплексный тип	F E	+	+	+	+	+
COMMAND_ARGUMENT_ COUNT ( )	Число аргументов в командной строке	F T	–	–	–	+	+
CONJG (Z)	Сопряженное комплексное число	F E	+	+	+	+	+
COS (X)	Функция косинус	F E	+	+	+	+	+
COSH (X)	Гиперболический косинус	F E	+	+	+	+	+
COUNT (MASK [, DIM] [, KIND**])	Число элементов со значением "истина”	F T	–	+	+	+	+
CPU_TIME (TIME)	Процессорное время	S	–	–	+	+	+
CSHIFT (ARRAY, SHIFT [, DIM])	Циклический сдвиг элементов массива	F T	–	+	+	+	+
DATE_AND_TIME ([DATE] [, TIME] [, ZONE] [, VALUES])	Дата и время	S	–	+	+	+	+
DBLE (A)	Преобразование к вещественному типу двойной точности	F E	+	+	+	+	+
DIGITS (X)	Число значащих цифр в модельном представлении X	F I	–	+	+	+	+
DIM (X, Y)	Положительная разность max(X-Y, 0)	F E	+	+	+	+	+
DOT_PRODUCT (VECTOR_A, VECTOR_B)	Скалярное произведение векторов	F T	–	+	+	+	+
DPROD (X, Y)	Произведение двух вещественных, результат - двойной точности	F E	+	+	+	+	+
DSHIFTL (I, J, SHIFT)	Комбинированный сдвиг влево	F E	–	–	–	–	+
DSHIFTR (I, J, SHIFT)	Комбинированный сдвиг вправо	F E	–	–	–	–	+
EOSHIFT (ARRAY, SHIFT [, BOUNDARY] [, DIM])	Нециклический сдвиг элементов массива	F T	–	+	+	+	+

EPSILON (X)	Число, почти не отличающееся от 1 для модели аргумента	F I	–	+	+	+	+
ERF (X) ERFC (X) ERFC_SCALED (X)	Функции ошибки	F E	–	–	–	–	+
EXECUTE_COMMAND_ LINE (COMMAND [, WAIT] [, EXITSTAT] [, CMDSTAT] [, CMDMSG])	Выполнение командной строки	S	–	–	–	–	+
EXP (X)	Экспонента	F E	+	+	+	+	+
EXPONENT (X)	Степенная часть модельного представления аргумента (порядок)	F E	–	+	+	+	+
EXTENDS_TYPE_OF (A, MOLD)	Проверяет, является ли динамический тип A расширением динамического типа MOLD	F I	–	–	–	+	+
FINDLOC (ARRAY, VALUE [, DIM] [, MASK] [, KIND] [, BACK])	Местоположение специфицированного значения	F T	–	–	–	–	+
FLOOR (A [, KIND*])	Наибольшее целое, меньшее или равное аргументу	F E	–	+	+	+	+
FRACTION (X)	Дробная часть модельного представления аргумента	F E	–	+	+	+	+
GAMMA (X)	Гамма-функция	F E	–	–	–	–	+
GET_COMMAND ([COMMAND] [, LENGTH] [, STATUS])	Возвращает командную строку	S	–	–	–	+	+
GET_COMMAND_ARGUMENT (NUMBER [, VALUE] [, LENGTH] [, STATUS])	Возвращает один аргумент командной строки	S	–	–	–	+	+
GET_ENVIRONMENT_VARIA BLE (NAME [, VALUE] [, LENGTH] [, STATUS] [, TRIM_NAME])	Выдает значение переменной окружения	S	–	–	–	+	+
HUGE (X)	Наибольшее число в модели представления аргумента	F I	–	+	+	+	+
HYPOT (X, Y)	Расстояние между двумя точками в Евклидовом пространстве	F E	–	–	–	–	+

IACHAR (C [, KIND**])	Позиция символа в кодировке ASCII	F E	–	+	+	+	+
IALL (ARRAY [, DIM] [,MASK])	Редукция массива посредством выполнения операции AND	F T	–	–	–	–	+
IAND (I, J)	Логическое AND (поразрядное)	F E	–	+	+	+	+
IANY (ARRAY [, DIM] [,MASK])	Редукция массива посредством выполнения операции OR	F T	–	–	–	–	+
IBCLR (I, POS)	Обнуление бита POS	F E	–	+	+	+	+
IBITS (I, POS, LEN)	Выделение последовательности битов	F E	–	+	+	+	+
IBSET (I, POS)	Установка бита POS равным единице	F E	–	+	+	+	+
ICHAR (C [, KIND**])	Номер позиции символа C в кодировке процессора	F E	+	+	+	+	+
IEOR (I, J)	Исключающее OR для битов	F E	–	+	+	+	+
IMAGE_INDEX (COARRAY, SUB)	Конвертирование коиндексов в индекс экземпляра	F I	–	–	–	–	+
INDEX (STRING, SUBSTRING [, BACK] [, KIND**])	Начальная позиция подстроки в строке символов	F E	+	+	+	+	+
INT(A [, KIND])	Преобразование в целый тип	F E	+	+	+	+	+
IOR (I, J)	Включающее OR для битов	F E	–	+	+	+	+
IPARITY (ARRAY [, DIM] [, MASK])	Редукция массива посредством выполнения операции исключающее OR	F T	–	–	–	–	+
ISHFT (I, SHIFT)	Логический сдвиг для битов	F E	–	+	+	+	+
ISHFTC (I, SHIFT [, SIZE])	Циклический сдвиг набора битов	F E	–	+	+	+	+
IS_CONTIGUOUS (ARRAY)	Проверка, является ли смежным размещение массива в памяти	F I	–	–	–	–	+
IS_IOSTAT_END (I)	Проверяет значение состояния “конец файла”	F E	–	–	–	+	+
IS_IOSTAT_EOR (I)	Проверяет значение состояния “конец записи”	F E	–	–	–	+	+

KIND (X)	Значение параметра разновидности типа	F I	–	+	+	+	+
LBOUND (ARRAY [, DIM] [, KIND**])	Нижние границы измерений массива	F I	–	+	+	+	+
LCOBOUND (COARRAY [, DIM] [, KIND])	Нижние когранницы комассива	F I	–	–	–	–	+
LEADZ (I)	Число ведущих нулей в битовом представлении числа	F E	–	–	–	–	+
LEN (STRING[, KIND**])	Длина символьного аргумента	F I	+	+	+	+	+
LEN_TRIM (STRING [, KIND**])	Возвращает длину строки за вычетом конечных пробелов	F E	–	+	+	+	+
LGE (STRING_A, STRING_B)	Лексически больше или равно в последовательности ASCII	F E	+	+	+	+	+
LGT (STRING_A, STRING_B)	Лексически больше в последовательности ASCII	F E	+	+	+	+	+
LLE (STRING_A, STRING_B)	Лексически меньше или равно в последовательности ASCII	F E	+	+	+	+	+
LLT (STRING_A, STRING_B)	Лексически меньше в последовательности ASCII	F E	+	+	+	+	+
LOG (X)	Натуральный логарифм	F E	+	+	+	+	+
LOG_GAMMA (X)	Логарифм абсолютного значения гамма функции	F E	–	–	–	–	+
LOG10 (X)	Десятичный логарифм	F E	+	+	+	+	+
LOGICAL (L [, KIND])	Преобразование объектов логического типа с разными параметрами типа	F E	–	+	+	+	+
MASKL(I [, KIND]) MASKR (I [, KIND])	Выравнивание маски	F E	–	–	–	–	+
MATMUL (MATRIX_A, MATRIX_B)	Умножение матриц, матрицы на вектор или вектора на матрицу	F T	–	+	+	+	+
MAX (A_1, A_2 [, A_3,…])	Максимальное значение	F E	+	+	+	+	+
MAXEXPONENT (X)	Максимальная экспонента в модели представления X	F I	–	+	+	+	+

MAXLOC (ARRAY [, DIM*] [, MASK] [, KIND**] [, BACK***])	Положение максимального элемента в массиве	F T	–	+	+	+	+
MAXVAL (ARRAY [, DIM] [, MASK])	Значение максимального элемента массива	F T	–	+	+	+	+
MERGE (TSOURCE, FSOURCE, MASK)	Слияние массивов под управлением маски	F E	–	+	+	+	+
MERGE_BITS (I, J, MASK)	Слияние битов под управлением маски	F E	–	–	–	–	+
MIN (A_1, A_2 [, A_3,…])	Минимальное значение	F E	+	+	+	+	+
MINEXPONENT (X)	Минимальная экспонента в модели представления X	F I	–	+	+	+	+
MINLOC (ARRAY [, DIM*] [, MASK] [, KIND**] [, BACK***])	Положение минимального элемента в массиве	F T	–	+	+	+	+
MINVAL (ARRAY [, DIM] [, MASK])	Значение минимального элемента массива	F T	–	+	+	+	+
MOD (A, P)	Функция остатка, т.е. A-INT(A/P)*P	F E	+	+	+	+	+
MODULO (A, P)	Функция “по модулю”	F E	_	+	+	+	+
MOVE_ALLOC (FROM, TO)	Передача размещения объекта	S	–	–	–	+	+
MVBITS (FROM, FROMPOS, LEN, TO, TOPOS)	Копирование битов	S E	–	+	+	+	+
NEAREST (X, S)	Ближайшее, отличное от аргумента процессорное число в направлении, определяемом знаком S	F E	–	+	+	+	+
NEW_LINE (A)	Символ новой строки	F I	–	–	–	+	+
NINT (A [, KIND])	Целое ближайшее к вещественному аргументу	F E	+	+	+	+	+
NORM2 (X [, DIM])	L^2 норма массива	F T	–	–	–	–	+
NOT (I)	Логическое дополнение для битов	F E	–	+	+	+	+
NULL ([MOLD])	Функция статуса связанности указателя	F T	–	–	+	+	+
NUM_IMAGES ( )	Число экземпляров (images)	F T	–	–	–	–	+

PACK (ARRAY, MASK [, VECTOR])	Упаковка массива в вектор под управлением логической маски	F T	–	+	+	+	+
PARITY (MASK [, DIM])	Редукция массива посредством выполнения операции NEQV	F T	–	–	–	–	+
POPCNT (I)	Число бит в битовом представлении аргумента	F E	–	–	–	–	+
POPPAR (I)	Четность, выраженная как 0 или 1.	F E	–	–	–	–	+
PRECISION (X)	Десятичная точность в модельном представлении X	F I	–	+	+	+	+
PRESENT (A)	“Истина“, если необязательный аргумент присутствует, иначе – “ложь”	F I	–	+	+	+	+
PRODUCT (ARRAY [, DIM] [, MASK])	Произведение элементов массива	F T	–	+	+	+	+
RADIX (X)	Основание системы счисления модели представления X	F I	–	+	+	+	+
RANDOM_NUMBER (HARVEST)	Генератор псевдослучайных чисел	S	–	+	+	+	+
RANDOM_SEED ([SIZE] [, PUT] [, GET])	Инициализация генератора псевдослучайных чисел	S	–	+	+	+	+
RANGE (X)	Значение диапазона десятичной экспоненты	F I	–	+	+	+	+
REAL(A [, KIND])	Преобразование в вещественный тип	F E	+	+	+	+	+
REPEAT (STRING, NCOPIES)	Конкатенация копий аргумента STRING	F T	–	+	+	+	+
RESHAPE (SOURCE, SHAPE [,PAD] [, ORDER])	Изменение конфигурации массива	F T	–	+	+	+	+
RRSPACING (X)	Обратная величина относительного расстояния между числами модели в области X	F E	–	+	+	+	+
SAME_TYPE_AS (A, B)	Сравнение динамических типов	F I	–	–	–	+	+
SCALE (X, I)	Умножение вещественного числа на основание системы счисления, возведен-	F E	–	+	+	+	+

	ное в целую степень, определяемую аргументом I ($X*b^I$)						
SCAN (STRING, SET [, BACK] [, KIND**])	Номер позиции символа из набора SET в аргументе STRING	F E	–	+	+	+	+
SELECTED_CHAR_KIND (NAME)	Значение параметра типа для символьного аргумента	F T	–	–	–	+	+
SELECTED_INT_KIND (R)	Значение параметра разновидности целого типа в заданном диапазоне	F T	–	+	+	+	+
SELECTED_REAL_KIND ([P] [,R] [, RADIX***])	Значение параметра разновидности типа для вещественных с заданными точностью и диапазоном	F T	–	+	+	+	+
SET_EXPONENT (X, I)	Установить экспонентную часть числа модельного представления аргумента	F E	–	+	+	+	+
SHAPE (SOURCE [, KIND**])	Конфигурация массива или скаляра	F I	–	+	+	+	+
SHIFTA (I, SHIFT) SHIFTL (I, SHIFT) SHIFTR (I, SHIFT)	Сдвиги влево и вправо	F E	–	–	–	–	+
SIGN (A, B)	Передача знака	F E	+	+	+	+	+
SIN (X)	Функция синус	F E	+	+	+	+	+
SINH (X)	Гиперболический синус	F E	+	+	+	+	+
SIZE (ARRAY [, DIM] [, KIND**])	Число элементов массива	F I	–	+	+	+	+
SPACING (X)	Абсолютное расстояние между модельными числами области X	F E	–	+	+	+	+
SPREAD (SOURCE, DIM, NCOPIES)	Расширение массива путем добавления копий указанного измерения	F T	–	+	+	+	+
SQRT (X)	Корень квадратный	F E	+	+	+	+	+
STORAGE_SIZE (A [,KIND])	Размер памяти в битах	F I	–	–	–	–	+
SUM (ARRAY [, DIM] [, MASK])	Сумма элементов массива	F T	–	+	+	+	+

SYSTEM_CLOCK ([COUNT] [, COUNT_RATE] [, COUNT_MAX])	Время системных часов	S	–	+	+	+	+
TAN (X)	Тангенс	F E	+	+	+	+	+
TANH (X)	Гиперболический тангенс	F E	+	+	+	+	+
THIS_IMAGE ( )	Индекс вызывающего экземпляра	F T	–	–	–	–	+
THIS_IMAGE (COARRAY [, DIM])	Коиндексы для этого экземпляра	F T	–	–	–	–	+
TINY (X)	Наименьшее положительное число в модельном представлении аргумента	F I	–	+	+	+	+
TRAILZ (I)	Число концевых нулевых бит	F E	–	–	–	–	+
TRANSFER (SOURCE, MOLD [, SIZE])	Передача типа	F T	–	+	+	+	+
TRANSPOSE (MATRIX)	Транспонирование матрицы	F T	–	+	+	+	+
TRIM (STRING)	Удаление конечных пробелов	F T	–	+	+	+	+
UBOUND (ARRAY [, DIM] [, KIND**])	Верхние границы измерений массива	F I	–	+	+	+	+
UCOBOUND (COARRAY [, DIM] [, KIND])	Верхние кограницы комассива	F I	–	–	–	–	+
UNPACK (VECTOR, MASK, FIELD)	Распаковка вектора в массив под управлением логической маски	F T	–	+	+	+	+
VERIFY (STRING, SET [, BACK] [, KIND**])	Положение символа из STRING, которого нет в SET	F E	–	+	+	+	+

Заключение

Сравнение стандартов языка Фортран показывает, что один из первых языков программирования претерпел существенное обновление.

Современный Фортран позволяет создавать мобильные и надежные программы, хорошо структурированные, наглядные и лаконичные. Язык содержит элементы непроцедурности, средства явной спецификации векторных операций, большой набор встроенных математических функций, развитые средства декомпозиции программ. Современный Фортран содержит средства поддержки объектно-ориентированного программирования, средства параллельного программирования и многие другие полезные черты.

Работа по развитию и совершенствованию Фортрана продолжается. Следующий стандарт будет содержать небольшое число нововведений. Основные новшества касаются дальнейшего развития средств поддержки параллельных вычислений.

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. С.Г. Дробышевич, редактор перевода А.М.Горелик. М.: Финансы и статистика, 1998
8. Горелик А.М Программирование на современном Фортране. – М.: Финансы и статистика, 2006
9. Горелик А.М. Эволюция языка программирования Фортран (1957–2007) и перспективы его развития. // Вычислительные методы и программирование, 2008, т.9, №2, с.223-241.
10. Gorelik A.M. Statements, data types and intrinsic procedures in the Fortran Standards. // ACM SIGPLAN Fortran Forum, 2014, №3, pp. 5-17.
11. Горелик А.М. Смешанное программирование на Фортране и Си. // Препринт ИПМ РАН, 2009, №59.
12. Горелик А.М. Средства поддержки параллельных вычислений в стандартах языка Фортран // Информационные технологии и вычислительные системы, 2014, №3, с.17-23.

Горелик Алла Моисеевна. Старший научный сотрудник ИПМ им. М.В. Келдыша РАН. Окончила Московский областной педагогический институт (физмат факультет). Кандидат физико-математических наук. Член ISO/IEC JTC1/SC22/WG5. Автор более 100 печатных работ и пяти книг. Область научных интересов: языки программирования и компиляторы, технологии программирования больших вычислительных задач, параллельные вычисления. E-mail: gorelik@keldysh.ru