УДК 681.3.06

СОВРЕМЕННЫЙ ФОРТРАН ДЛЯ КОМПЬЮТЕРОВ ТРАДИЦИОННОЙ АРХИТЕКТУРЫ И ДЛЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

А. М. Горелик[1]

В статье представлен аналитический обзор новых возможностей действующего международного стандарта языка Фортран (Фортран 95); рассматриваются проблемы стандартизации языка и перспективы его дальнейшего развития. Наибольшее внимание уделяется тем новым средствам языка, которые позволяют использовать современные технологии программирования, а также средствам поддержки параллельности, которые имеются в действующем стандарте.

Предисловие. Исследования, связанные с высокопроизводительными вычислениями, обычно направлены на решение важной проблемы повышения эффективности выполнения программ на современных компьютерах с параллельной архитектурой.

Другая серьезная проблема заключается в том, чтобы повысить эффективность разработки больших и сложных вычислительных программ, т.е. уменьшить время разработки и облегчить труд программиста. Для решения таких проблем используются современные технологии программирования.

При программировании сложных вычислительных задач для современных высокопроизводительных ЭВМ лидирующее положение занимает язык Фортран. Однако старые варианты языка (Фортран 77 и более ранние версии), которые еще часто используются, не позволяют применять современные методы программирования.

В современный Фортран (Фортран 90 и Фортран 95 [1 – 3]) введены средства, которые позволяют использовать новые технологии. Поскольку практически сохраняется преемственность с предыдущими стандартами языка, можно использовать ранее созданные прикладные программы, и некоторые из новых концепций могут быть применены для модификации старых программ.

По сравнению с предыдущими версиями языка, Фортран 90/95 обеспечивает современный стиль программирования, позволяет создавать более мобильные и более надежные программы, лучше структурированные, а потому более наглядные и более лаконичные. Кроме того, современный Фортран содержит новые средства декомпозиции программ, которые особенно полезны для разработки больших вычислительных программ, так как облегчают разделение работы между исполнителями таким образом, чтобы программа состояла из как можно более независимых частей и интерфейс между ними был бы удобным для разработчиков.

Для параллельного программирования вычислительных задач, безусловно, целесообразно использовать именно современный Фортран. Фортран 90 содержит некоторые средства параллельной обработки, в Фортране 95 средства поддержки параллельности получили дальнейшее развитие. Кроме того, разработанные системы для параллельного программирования ориентируются на новые средства, которые введены в современный Фортран.

Статья посвящена анализу современного состояния и перспективам развития языка Фортран; она представляет собой переработанный и дополненный вариант первой части аналитического обзора, который в 2002 г. был поддержан Российским фондом фундаментальных исследований (проект 02–01–07012).

В статье анализируются новые (по сравнению с Фортраном 77) возможности действующего международного стандарта (Фортран 95), рассматриваются проблемы стандартизации языка и перспективы его дальнейшего развития. Наибольшее внимание уделяется тем новым средствам языка, которые позволяют использовать современные технологии программирования, а также средствам поддержки параллельности, которые имеются непосредственно в действующем стандарте.

В дальнейшем предполагается опубликовать обзор языков и систем для параллельного программирования, ориентированных на современный Фортран.

1. Современный международный стандарт языка Фортран. Стандартизация языков программирования создает предпосылки для повышения мобильности программного обеспечения. Высокая

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

мобильность облегчает адаптацию программы для работы в различных окружениях, что позволяет использовать ранее созданные прикладные программы и облегчает перенос программы с одной платформы на другую.

Язык Фортран подвергался стандартизации в рамках ANSI и ISO четыре раза (Фортран 66, Фортран 77, Фортран 90, Фортран 95). Действующий международный стандарт языка был принят в 1997 г.; неформальное название его — Фортран 95 [1]. Язык является относительно небольшим расширением предыдущего стандарта — Фортран 90 [2, 3], который был утвержден в 1991 г. и пока не потерял своей актуальности.

В настоящее время опубликован проект будущего стандарта, неформально названного Фортран 2003, который предусматривает весьма существенные нововведения. Утверждение стандарта планируется в конце 2004 г.

Современные стандарты Фортрана представляют собой семейство стандартов, состоящее из нескольких частей. Первая часть — основной (базовый) язык [1]. Остальные части являются дополнительными. При этом не требуется, чтобы компилятор, соответствующий базовому стандарту, обязательно реализовывал дополнительные части. Вторая часть стандарта [4] содержит описание средств для работы с символьными строками переменной длины. Третья часть определяет описание языка условной компиляции [5].

Международные стандарты языка являются результатом совместной деятельности экспертов многих стран. Стандартизацией языка Фортран занимаются Американский технический комитет J3 ANSI и эксперты рабочей группы ISO/IEC JTC1/SC22/WG5 (WG5). Членами WG5 являются специалисты ряда стран, в т.ч. и нашей страны. В их числе представители компьютерных фирм, крупных университетов. Многие из тех, кто ответствен за разработку коммерческих Фортран-компиляторов, являются членами J3 и/или WG5.

В нашей стране роль стандартов на языки программирования недооценивается как разработчиками прикладных программ, так и разработчиками системного программного обеспечения, и новое программное обеспечение часто создается без учета международных стандартов. Это затрудняет адаптацию программ к другой вычислительной среде.

Официальное описание стандарта — строго формализованное. Для читателя, особенно не очень подготовленного, более понятной является учебная и методическая литература. Выпущены десятки книг на разных (более чем на десяти) языках, которые содержат неформальное описание языков Фортран 90 и Фортран 95 (например, [6, 7]). В различных университетах мира читаются лекции, организованы разнообразные курсы по изучению современного Фортрана. На русском языке книг, содержащих описание стандарта языка Фортран 95, пока нет. Для Фортрана 90 выпущен перевод официального описания стандарта [3] и книги [8, 9] с неформальным описанием этого стандарта.

В работах [10 – 12] имеется более полная информация о целях стандартизации и о том, кто и как разрабатывает международные стандарты.

Фортран 90/95 реализован практически на всех современных ЭВМ для различных платформ. Разработчики компиляторов предлагают современный набор инструментов (включая средства отладки, средства визуализации и др.) для разработки Фортран-приложений. Созданы и другие полезные программные продукты, связанные с современным Фортраном (анализаторы, конверторы, переводящие программу с Фортрана 77 на современный Фортран, разнообразные системы тестов, графические библиотеки и др.). Разработаны также математические библиотеки, в том числе и для параллельной архитектуры (например, IMSL, NAG и др.).

Информацию о публикациях, реализациях и другие полезные сведения, касающиеся современного Фортрана, можно найти в Интернете (см., например, [13]).

В следующих разделах приводится краткий обзор новых по сравнению с Фортраном 77 средств базового языка. Там, где специально не оговорено, эти новшества имеются и в Фортране 90, и в Фортране 95.

2. Формат исходного текста. В языке имеется две формы записи исходного текста: свободный формат (современная форма) и фиксированный формат (устаревшая форма). В свободном формате можно помещать более одного оператора в строке, при этом в качестве разделителя используется точка с запятой. Признак продолжения оператора на строку продолжения — символ & — указывается в конце той строки, которую надо продолжить. Метка оператора в свободном формате записывается перед оператором. Комментарии записываются после символа "восклицательный знак" в начале строки или в любой позиции строки после оператора. В свободном формате пробелы являются значащими. Они должны отделять смежные лексические единицы.

Имена могут содержать до 31 символа: латинские буквы (строчные и прописные), цифры и сим-

вол подчеркивания. Строчные буквы всюду, кроме символьных констант и символьных спецификаций формата, считаются равнозначными соответствующим прописным буквам.

Расширен набор символов. В частности, введены более удобные обозначения для операций отношения ($==, <, <=, >, >=, /=$).

3. Типы и атрибуты данных. В Фортране 90/95 каждый объект данных имеет тип, ранг (размерность), а также может иметь некоторое количество дополнительных свойств — атрибутов. Эти свойства определяют характеристики соответствующего объекта и особенности его использования.

Свойства объектов могут быть специфицированы двумя способами: в операторах спецификации атрибутов (как и прежде), либо с помощью атрибутов в операторе объявления типа. В одном операторе объявления типа можно описать несколько или все атрибуты объектов, перечисленных в данном операторе после двойного двоеточия; в этих операторах допускается также инициализация объектов.

Пример.

```
! Новая форма объявления типа
INTEGER, DIMENSION (5), PARAMETER :: ARRAY = (/ 1, 2, 3, 4, 5 /)
```

Имеются традиционные для Фортрана встроенные (предопределенные) типы данных и средства, позволяющие определять в программе производные типы данных (см. раздел 4).

Встроенные типы данных: логический, целый, вещественный, комплексный и символьный. Для всех встроенных типов имеются средства параметризации, т.е. каждый встроенный тип может иметь несколько разновидностей. Параметризация встроенных типов обеспечивает возможность задания способа представления данных, что позволяет компиляторам поддерживать короткие целые, более двух видов точности для вещественных и комплексных данных, а также многобайтовые символьные данные (для языков с большим набором символов) или использовать дополнительные наборы символов для конкретных приложений. Наличие таких средств повышает мобильность программы, т.к. облегчает адаптацию программы к конкретному окружению (подробнее см. в [14]).

Пример.

```
! Два варианта для задания способа представления, обеспечивающего точность
! не менее 10 знаков и диапазон десятичного порядка не менее чем (-30, +30)
INTEGER, PARAMETER :: PARAM = SELECTED_REAL_KIND (10, 30)
REAL (KIND = PARAM) X, Y   ! PARAM — параметр типа
REAL (SELECTED_REAL_KIND(10, 30)) :: P, Q
```

Для языка Фортран традиционно был характерен принцип умолчания, т.е. автоматическое приписывание объектам типа, если тип объекта не указан явно. В Фортран 90/95 введен оператор IMPLICIT NONE. При наличии такого оператора типы объектов в данной программной единице должны быть объявлены явно. Явное объявление объектов обеспечивает возможность контроля типов, что, очевидно, полезно для повышения надежности создаваемых программ. IMPLICIT NONE является более гибким инструментом, чем просто обязательность описаний, что имеет место в других языках.

4. Типы и операции, определяемые пользователем. В Фортране 90/95 имеются средства, позволяющие определить новые типы данных — производные типы (структуры данных), которые представляют собой совокупность компонент. Компоненты производного типа могут иметь встроенный или ранее описанный производный тип; компонентами могут быть скаляры, массивы и указатели. Компонент производного типа может иметь такой же тип, как и описываемый, если он является указателем. Это полезно для создания связанных списков. Объекты производного типа объявляются в операторах с ключевым словом TYPE; они могут быть скалярами и массивами.

Компоненты объекта производного типа могут использоваться в обычных выражениях и операторах. Для объектов одинакового производного типа определена операция присваивания.

В языке имеется аппарат, позволяющий программисту распространить для производных типов встроенные операции (перегрузка) или описать новые операции и присваивания для данных встроенных и производных типов.

Операции описываются с помощью функций; если такая функция имеет один формальный аргумент, она описывает унарную операцию, если два — бинарную. Эти операции можно затем использовать в выражениях. Определяемый оператор присваивания описывается с помощью подпрограммы, имеющей два аргумента, которые соответствуют левой и правой частям присваивания.

Примеры.

```
! Описание типа EMPLOYEE
TYPE EMPLOYEE
  CHARACTER (LEN = 15) NAME
  INTEGER SSN
  REAL SALARY
END TYPE EMPLOYEE

! Объявление объектов описанного типа
TYPE (EMPLOYEE) :: IVANOV, PETROV, SIDOROV

! Присваивание компоненту объекта производного типа
PETROV%SSN = 12345
```

Такие средства позволяют программисту определить абстрактные типы данных, что является одним из элементов объектно-ориентированного программирования (см. раздел 9; более подробно в [15, 16, 18, 19]).

5. Операции над массивами, секциями массивов и соответствующие присваивания. Операторы и конструкции WHERE и FORALL. Язык предоставляет широкий набор средств для работы с массивами и секциями массивов как с целыми объектами на поэлементной основе. Эти средства обеспечивают большие возможности для упрощения организации и написания почти всех программ, использующих массивы, т.к. позволяют во многих случаях обойтись без вложенных циклов и условных переходов. Многие алгоритмы, включающие работу с массивами, могут быть записаны лаконичнее и нагляднее как последовательность вычислений над полными массивами. Такие средства делают программу более структурной и более объектно-ориентированной. Программу, содержащую такие средства, легче отлаживать, модифицировать и распараллеливать.

Помимо указанных выше преимуществ, существенным является тот факт, что операции над массивами фактически специфицируют параллелизм действий над компонентами массивов, поскольку явно специфицируют векторные операции и параллельные циклы. Это позволяет компилятору сгенерировать эффективный код с учетом конкретной архитектуры, особенно для тех современных ЭВМ, которые имеют аппаратные средства векторной обработки. Эти свойства аппаратуры, в свою очередь, формировались на основе требований решаемых задач.

В данном разделе приводятся краткие сведения об имеющихся средствах для работы с массивами.

В языке разрешены выражения и присваивания для массивов. Например, если A и B — двумерные массивы, то операция A*B означает не умножение матриц (для этого имеется встроенная функция MATMUL), а поэлементное умножение, выполняемое в произвольном порядке.

Секция массива позволяет адресоваться к части массива; она задается с помощью имени массива и списка индексов секции. Конструкторы массивов позволяют задавать последовательность скаляров в качестве одномерного массива без имени.

Имена массивов без индексов (полные массивы), секции массивов и конструкторы массивов могут использоваться в качестве операндов выражений, в левой части оператора присваивания, как фактические аргументы процедур, в списке ввода-вывода. Кроме того, массив может быть также значением выражения и значением функции (как встроенной, так и определяемой пользователем).

Имеется также возможность условного присваивания массиву под управлением логической маски, организации параллельного цикла, большой набор встроенных процедур для работы с массивами и средства динамического размещения массивов.

Примеры секций и конструктора массива. Пусть имеется объявление:

```
INTEGER, DIMENSION :: Z(M,N), V(4), A(3)
```

Тогда Z (3, :) — третья строка матрицы Z; Z (:, J) — J-й столбец матрицы Z; V = (/ 2, 1, 1, 3 /) — конструктор массива; A (3, V) — секция массива, содержащая элементы A (3, 2), A (3, 1), A (3, 1) и A (3, 3).

Рассмотрим более сложный пример. Секция X (:, 1:N, 4, 3:1:-1) содержит целиком первое измерение, позиции с 1 по N второго измерения, позицию 4 третьего измерения и позиции с 1 по 3 из четвертого измерения в обратном порядке. Это искусственный пример, выбранный для иллюстрации разнообразия форм. Конечно, более частым применением может быть выбор строки или столбца из массива, как в первых двух примерах.

При использовании массивов в выражениях должны соблюдаться некоторые правила. Операнды,

участвующие в одной операции, должны быть согласованы по конфигурации (т.е. должны иметь одинаковую размерность и размер по каждому измерению; скаляр считается согласованным с любым массивом). Значение таких выражений также является массивом. Поэлементное выполнение операций и присваиваний может производиться параллельно, однако результат не зависит от того, поддерживает ли компилятор параллельное выполнение.

Примеры.

```
REAL, DIMENSION (0 : 9) :: A
REAL, DIMENSION (5 : 14) :: C
REAL, DIMENSION (1 : 20, 1 : 20) :: X, Y, Z
! Первым 10 элементам первого столбца Y присваиваются значения A
Y(1 : 10, 1) = A
! Элементы J-й строки матрицы X присваиваются I-му столбцу матрицы Y
Y(1 : 20, I) = X(J, 1 : 20)
! Первые M элементов J-й строки матрицы Y
! присваиваются первым M элементам I-го столбца матрицы X
X(1:M, I) = Y(J, 1:M)
C = A(9:0:-1)
! Элементы с 2-го по 4-й передвигаются и становятся элементами с 3-го по 5-й
A(3:5) = A(2:4)
```

В Фортране 77 подобные примеры могут быть реализованы только с использованием циклов, для последнего примера требуется еще и временный массив. Очевидно, что запись на современном Фортране более короткая и более наглядная.

Оператор присваивания по маске и конструкция присваивания по маске позволяют, в зависимости от некоторого условия, выполнить вычисление выражения и присваивание значений не для всех элементов массива, т.е. выполнить присваивание под управлением логической маски. Например, чтобы присвоить нуль всему массиву, достаточно написать A=0.0; чтобы присвоить нуль только отрицательным элементам, потребуется записать условное присваивание

```
WHERE (A < 0.0) A = 0.0
```

Конструкция WHERE—END WHERE, как и другие конструкции языка (см. раздел 10), отличается от обычных операторов тем, что внутри нее могут помещаться другие операторы или блоки операторов, рассматриваемые как единое целое.

Пример конструкции WHERE.

```
WHERE (W /= Y)
  X(10, :) = (W + Y)/(W − Y)
ELSEWHERE
  X(10, :) = 0.0
END WHERE
```

В Фортране 95 введены некоторые расширения конструкции WHERE: оператор ELSEWHERE может иметь маску; конструкция WHERE может быть вложенной; конструкция может иметь имя.

```
! Пример на Фортране 95
EXAMPLE: WHERE (X > 0.0)
           X = X + 1.0
         ELSEWHERE (X < 0.0)
           X = X + 2.0
         ELSEWHERE
           X = X + 3.0
         END WHERE EXAMPLE
```

Множественное присваивание массиву обеспечивается не только с помощью описанных выше средств, но также с помощью оператора и конструкции FORALL, которые введены в Фортран 95 (в Фортране 90 они отсутствовали).

Пример.

```
! Присваивание диагонали матрицы A
FORALL (I = 1:N) A (I, I) = B(I)
```

Одно из отличий от WHERE заключается в том, что FORALL использует элементы массивов, а WHERE ориентируется на целый массив. FORALL допускает возможность специфицировать больший класс секций массивов, чем допускалось в Фортране 90. Так, в приведенном выше примере представлена диагональ матрицы; с помощью средств Фортрана 90 можно представить только прямоугольные секции.

FORALL похож на DO-циклы, но выполняется иначе — сначала вычисляются правые части для всех значений индексов, а затем производятся присваивания. Выполнение (для различных значений индексов) может быть параллельным, но результат не зависит от того, поддерживает ли компилятор параллельное выполнение FORALL.

Так, результат выполнения оператора

$$\text{FORALL (I} = 2 : \text{N) C(I, I)} = \text{C(I} - 1, \text{ I} - 1)$$

отличен от результата следующей последовательности операторов

```
DO I = 2, N
  C(I, I) = C(I − 1, I − 1)
END DO
```

Различие объясняется тем, что итерации DO-цикла выполняются последовательно; в приведенном примере каждый следующий элемент диагонали модифицируется до его использования в следующей итерации. В противоположность этому, в FORALL все диагональные элементы выбираются и используются до сохранения модифицированного значения в памяти.

Помимо оператора FORALL, имеется также конструкция FORALL—END FORALL. Заголовок конструкции может содержать логическую маску.

Пример конструкции FORALL с маской.

```
FORALL (I = 1 : N, Y(I).NE.0.0)
  X(I) = 1.0/Y(I)
END FORALL
```

Конструкция FORALL допускает вложенность и может быть вложена в конструкцию WHERE, и наоборот, конструкция WHERE может быть вложена в FORALL. Конструкция может иметь имя, которое позволяет ее идентифицировать. Имена особенно полезны для удобства написания и чтения программ, содержащих вложенные конструкции.

Помимо базовых элементов и конструкций для работы с массивами в языке имеется большой набор встроенных функций, которые также существенно упрощают программирование вычислительных задач. Это объясняется тем, что часто используемые математические функции входят в состав самого языка, а не программируются каждым пользователем заново. Поскольку такие функции реализуются компилятором, они ориентируются на конкретную архитектуру и потому реализуются эффективно.

Имеется три категории таких функций:

— поэлементные функции, которые допускают обращение как со скалярным аргументом, так и с аргументом-массивом, и выполняются на поэлементной основе;

— справочные функции, результат которых зависит от свойств фактического аргумента, а не от его значения; такие функции позволяют получить в динамике информацию о свойствах массива;

— функции преобразования массивов имеют один или несколько формальных аргументов-массивов и часто имеют результатом массив; это следующие группы функций: функции редукции, функции умножения векторов и матриц, функция транспонирования матрицы, функция слияния массивов под управлением маски, функции упаковки и распаковки массивов; функция конструирования массива добавлением копий из элементов исходного массива; функция изменения конфигурации массива, функции сдвига, функции определения положения в массиве.

Примеры.

```
REAL, DIMENSION (1 : 20, 1 : 20) :: X, Y, Z
INTEGER NF, NC, K, N
Z = EXP(X + Y) − EXP(X − Y)
!  N факториал, т.е. 1*2*...*N
NF = PRODUCT ((/ (K, K = 2, N) /))
!  Число отрицательных элементов массива X
NC = COUNT (X, X < 0.0)
```

В Фортране 95 введены средства, позволяющие использовать не только встроенные поэлементные функции (как в Фортране 90), но также и поэлементные процедуры (функции и подпрограммы), определяемые пользователем. Такие процедуры должны иметь в заголовке префикс ELEMENTAL.

Более полная информация о средствах, описанных в этом разделе, имеется (помимо официальных описаний стандартов [1 – 3]) в работах [6 – 9, 17].

6. Механизмы динамического размещения массивов. В языке имеются различные механизмы динамического размещения массивов, которые весьма удобны, а во многих случаях и необходимы для задач, оперирующих с большими массивами. В предыдущих стандартах языка Фортран не разрешалось использовать массивы, границы которых вычисляются в процессе выполнения программы.

В современных стандартах введены следующие механизмы динамического размещения массивов.

1) Автоматически размещаемые массивы — массивы (не формальные аргументы), границы которых вычисляются при входе в процедуру; такие массивы являются локальными, они используются только внутри процедуры (пример 1).

2) Динамически размещаемые массивы — массивы, границы которых вычисляются при выполнении программы. Такие массивы объявляются с атрибутом ALLOCATABLE; при объявлении указывается только ранг массива (число двоеточий). При выполнении оператора ALLOCATE вычисляются границы массива, его размер и отводится память. Память, занимаемая массивом, освобождается при выполнении оператора DEALLOCATE (пример 2).

3) Массивы, создаваемые в процессе выполнения программы, т.е. массивы, которые не были заранее явно объявлены. Для создания таких массивов используется оператор ALLOCATE, содержащий указатель (см. раздел 7). При выполнении такого оператора создается динамический объект, определяются его границы, объекту отводится область памяти, указатель связывается с этим объектом и может в дальнейшем использоваться для ссылки на него. При выполнении оператора DEALLOCATE освобождается память от тех динамических объектов, на которые ссылаются указатели, перечисленные в данном операторе. Создаваемые объекты не имеют имени, ссылаться на них можно только с помощью указателей (пример 3).

Пример 1.

```
SUBROUTINE S (X, M)
REAL, DIMENSION (M) :: A   ! Автоматический массив A
```

Пример 2.

```
REAL, ALLOCATABLE :: Y(:, :)   ! Размещаемый массив
......
READ (*, *) M, N
ALLOCATE Y (M, N)   ! Размещение массива Y
```

Пример 3.

```
! Объявление указателя
REAL, POINTER :: A (:, :)
......
READ (*, *) N, M
! Создание и размещение динамического массива, связанного с указателем A
ALLOCATE (A (N, M))
```

7. Указатели. Важным новшеством языка являются указатели — объекты с атрибутом POINTER. С помощью указателей можно ссылаться на объекты; использование указателей и производных типов позволяет создавать произвольные структуры данных. Использование указателей позволяет также создавать в процессе выполнения программы динамические объекты (см. раздел 6).

При объявлении указателей специфицируются свойства (тип, размерность и т.п.) тех объектов, которые могут быть связаны с данным указателем; если указатель является массивом, объявляется только размерность, границы массива в описании опущены — они определяются только во время установления связи указателя с объектом-массивом.

Пример.

```
REAL, DIMENSION (:, :), POINTER :: X, Y
```

Указатель можно интерпретировать как дескриптор, который содержит всю информацию, необходимую для описания и размещения в памяти объекта того же типа, с теми же параметрами типа и той

же размерности, что и в объявлении указателя. При объявлении указателя отводится область памяти для дескриптора, но область памяти для самого объекта не отводится. Дескриптор сначала создается пустой и заполняется во время установления связи с объектом.

Связь указателя с объектом может быть установлена либо при выполнении оператора присваивания указателю, либо при выполнении оператора ALLOCATE, когда создается динамический объект.

После того как установлена связь указателя с объектом, указатель может появиться в любом месте, где может появиться обычный объект того же типа и с теми же параметрами типа, что и в объявлении указателя. Значением операнда-указателя является значение объекта, связанного с ним в данный момент.

Для присваивания значения собственно указателю служит уже упоминавшийся оператор присваивания указателю. Он устанавливает связь указателя с существующим объектом, оператор NULLIFY разрывает связь указателя с объектом.

Отметим, что в Фортране 90 не было способа определить начальный статус связанности для указателя, в Фортране 95 такая возможность появилась.

Примеры.

```
! Атрибут TARGET означает, что к массиву MATR можно обращаться
! с помощью указателя
REAL, TARGET :: MATR (M, N)
REAL, POINTER :: ROW (:), WINDOW (:, :)
ROW => MATR (M, :)   ! ROW связывается со строкой матрицы
! WINDOW связывается с секцией массива MATR
WINDOW => MATR (I − 1 : I + 1, J − 1 : J + 1)
```

Последний пример иллюстрирует еще одно полезное свойство указателя: в тех случаях, когда часто используется одна и та же секция массива, можно для краткости записи использовать указатель в качестве псевдонима.

8. Модули. Модуль — это новый вид программных единиц в Фортране. Программные единицы-модули предназначены для описания различных объектов (данных, производных типов, определяемых пользователем операций, процедур, интерфейсных блоков, namelist-списков), доступных другим программным единицам. Такие средства описания глобальных объектов являются более современными и более удобными, чем традиционный для Фортрана оператор COMMON (который используется только для данных и имеет ряд недостатков).

Объекты модуля, которые используются в других программных единицах, должны иметь атрибут PUBLIC (явно описанный или по умолчанию). Те объекты модуля, которые предназначены для использования лишь внутри данного модуля, должны быть объявлены с атрибутом PRIVATE.

В программной единице, использующей объекты некоторого модуля, должен присутствовать оператор USE с указанием имени модуля. Оператор USE позволяет ограничить доступ к общедоступным объектам и переименовать доступные. Переименование объектов бывает удобно, а иногда и необходимо, если в программной единице, использующей объекты модуля, имеются свои локальные объекты, имена которых совпадают с именами глобальных объектов модуля.

Использование модулей обеспечивает возможность зависимой раздельной компиляции программных единиц. Это объясняется тем, что информация, специфицируемая в модуле, доступна процессору при компиляции программных единиц, использующих данный модуль. В предыдущих стандартах языка предполагалась полностью независимая компиляция программных единиц, что не позволяло обнаружить некоторые ошибки на этапе компиляции. Зависимая и в то же время раздельная компиляция повышает надежность программ и позволяет компилятору выполнять более широкий спектр оптимизирующих преобразований.

Использование модулей для объектно-ориентированного программирования обсуждается в следующем разделе.

9. Поддержка объектно-ориентированного программирования. Предыдущие разделы уже позволяют читателю убедиться в том, что современный Фортран поддерживает значительную часть методологии объектно-ориентированного программирования.

Объектно-ориентированное программирование (ООП) — это современная методология программирования, предназначенная, в первую очередь, для разработки больших проектов. Основная идея ООП заключается в том, чтобы обеспечить разработчику возможность создавать типы данных, соответствующие понятиям прикладной задачи, вводить новые операции и структурировать программу таким образом, чтобы эти типы и операции были доступны извне, но детали реализации были бы скрыты от пользователя.

Использование механизмов ООП облегчает разделение работы между исполнителями при проектировании больших программ или комплекса программ, упрощает модификацию программы при изменении требований. Объектно-ориентированная программа обычно более короткая, более наглядная и более надежная.

Основные концепции ООП: расширяемость типов и операций, механизм инкапсуляции для реализации абстрактных типов данных, наследование, статический и динамический полиморфизм.

Наличие программных единиц-модулей, производных типов, операций, определяемых пользователем, и атрибутов PUBLIC и PRIVATE обеспечивает полную поддержку инкапсуляции и абстракции данных. Такие средства позволяют скрывать внутри модуля структуру типов и реализацию операций и позволяют их изменять в модуле без изменения программных единиц, использующих объекты этого модуля. Наличие различных механизмов динамического размещения объектов также обеспечивает поддержку инкапсуляции и абстракции данных.

ООП обеспечивает также возможность создавать многократно используемые компоненты программы без дублирования кодов с помощью механизма наследования, статического и динамического полиморфизма.

Наследование в Фортране 90/95 реализуется частично посредством производного типа и модулей; полное наследование в явном виде не поддерживается, но может быть смоделировано.

Статический полиморфизм позволяет использовать какой-либо объект более чем одним способом (причем конкретизация способа использования выполняется на этапе компиляции). Статический полиморфизм поддерживается в Фортране 90/95 следующим образом: явное (более надежно) и неявное преобразование типов — средство, традиционное для Фортрана; перегрузка операций и процедур, т.е. использование одного и того же имени для нескольких разных объектов в общей области видимости.

Динамический полиморфизм (динамическое связывание) в явном виде в Фортране 90/95 отсутствует, но может быть смоделирован.

Проблемы ООП для современного Фортрана более подробно обсуждаются в работах [15, 16, 18, 19].

В проекте будущего стандарта (Фортран 2003) предусмотрена полная поддержка ООП (см. раздел 14).

10. Поддержка структурного программирования. В Фортране 77 некоторые операторы управления противоречат концепции структурного программирования и сейчас признаны устаревшими (см. раздел 13). В отличие от него, Фортран 90/95 имеет полный комплект современных управляющих структур.

Наряду с операторами управления и конструкцией IF—END IF, введенной в Фортран 77, в современном Фортране имеется конструкция выбора SELECT CASE—END SELECT, которая позволяет выбрать для исполнения один из вариантов-блоков операторов в зависимости от значения некоторого выбирающего выражения. Введена также структурная конструкция цикла DO—END DO, в которой отсутствует метка; заголовок цикла может иметь не только традиционную форму (с управляющей переменной цикла), но и новую форму с ключевым словом WHILE или без управляющей информации; оператор EXIT вызывает выход из цикла на оператор, непосредственно следующий за заключительным оператором цикла; оператор CYCLE вызывает переход на следующую итерацию цикла. Использование этих двух операторов делает программу более наглядной и больше соответствует принципам структурного программирования, чем использование оператора GO TO и меток.

Введены имена конструкций, которые позволяют их идентифицировать; это особенно полезно при вложенных конструкциях, так как делает программу более наглядной. Имя конструкции не может использоваться для передачи управления.

Средства работы с массивами как с целыми объектами также служат улучшению структуры программы, так как во многих случаях позволяют обходиться без циклов и условных переходов (см. раздел 5).

Программу, которая хорошо структурирована, легче отлаживать, модифицировать и легче распараллеливать.

11. Процедуры. В современном Фортране существенно расширены средства, связанные с процедурами. В частности, с помощью процедур можно определить новые операции; можно определить родовые процедуры, рекурсивные, внутренние процедуры, специфицировать аргументы по назначению: INTENT(IN), INTENT(OUT), INTENT(INOUT). Имеется возможность использования необязательных и ключевых аргументов при вызове процедур, что позволяет перечислять их в произвольном порядке.

Введены средства задания явного интерфейса с помощью интерфейсного блока. Явный интерфейс в некоторых случаях является обязательным, так как позволяет правильно сгенерировать вызов процедуры; в других случаях использование интерфейсного блока позволяет проверить правильность вызовов

процедур (например, проверить соответствие типов аргументов). Использование явного интерфейса, как и спецификация аргументов по назначению, полезно для повышения надежности (подробнее о средствах повышения надежности см. в [14]).

Важным новшеством в Фортране 95 является возможность специфицировать функцию без побочного эффекта. Такие функции объявляются с ключевым словом PURE. Побочный эффект вызывает проблемы в параллельном программировании.

Новыми в Фортране 95 являются поэлементные процедуры, не только встроенные, как в Фортране 90, но и определяемые пользователем (см. раздел 5).

Помимо перечисленного, в языке имеется большой набор новых встроенных процедур (более 100 встроенных процедур: математических, справочных и др.). В Фортране 95 введены новые стандартные процедуры и расширены возможности некоторых процедур, введенных в Фортран 90.

12. Средства поддержки параллельности. В предыдущих разделах уже отмечались средства языка, которые могут использоваться для параллельного программирования. Фортран 95 не ориентирован на многопроцессорные системы. Для таких систем разработаны специальные языки, ориентированные на Фортран, которые существенно используют новые возможности современных стандартов языка.

В то же время, Фортран 90/95 содержит богатый набор средств, ориентированных на архитектуру с векторными операциями [17, 20]. В современные стандарты, как отмечалось выше (см. раздел 5), введены средства для работы с массивами и секциями массивов как с целыми объектами, векторные операции, стандартные процедуры для работы с массивами, поэлементные процедуры, условное присваивание массиву под управлением логической маски. Эти средства позволяют компилятору сгенерировать эффективный код с учетом особенностей аппаратуры.

В Фортран 95 дополнительно введены оператор и конструкция FORALL, которые позволяют специфицировать параллельный цикл (см. раздел 5). Введена спецификация PURE для функции без побочного эффекта. Вызов такой функции можно использовать в тех случаях, где возможна параллельная обработка без таких нежелательных последствий как недетерминизм. Поэлементные процедуры, встроенные и определяемые пользователем (см. разделы 5 и 11), также обеспечивают дополнительные возможности для выражения параллелизма.

Необходимость многих из указанных выше средств (помимо дополнительных удобств) вызвана появлением на современных ЭВМ аппаратных средств векторной обработки.

До появления этих средств в стандарте для таких компьютеров создавались векторизующие компиляторы, помогающие эффективно использовать возможности, заложенные в аппаратуре. Однако автоматическая векторизация (т.е. выявление скрытого параллелизма) требует довольно сложного анализа исходной программы, и при этом некоторые конструкции последовательных языков могут затруднить или даже сделать невозможной векторизацию. В некоторые системы программирования вводились специальные директивы, помогающие компилятору выполнять векторизацию исходной программы, были разработаны и языковые расширения. Достаточно полная информация об исследованиях проблем векторизации содержится в сборнике статей [21] и в статьях [22, 23]; отечественные разработки для ПС-3000 и ЕС1191 описаны в [24, 25].

Помимо перечисленных выше средств параллельности, Фортран содержит новые черты, которые, хотя и не связаны непосредственно с параллельностью, но используются в стандартных языках и системах параллельного программирования, ориентированных на Фортран (OpenMP, HPF, MPI): динамические и размещаемые массивы, указатели, производные типы данных, модули, явный интерфейс, определяемые пользователем операции, новые встроенные процедуры, операции над массивами, длинные имена и др.

13. Концепция эволюционного развития языка. В современном стандарте принята концепция дальнейшего эволюционного развития языка.

Некоторые элементы языка стали излишними после введения новых средств; такие элементы отнесены к категории устаревших черт. Разумеется, что если в язык добавлять новые средства без удаления уже ненужных, устаревших элементов, язык станет очень громоздким с большим числом дублирующих черт. В то же время, исключение из языка каких-либо элементов может затруднить использование ранее созданных программ. Учитывая эти противоречивые факторы, было решено заранее объявлять элементы языка, которые являются устаревшими, нерекомендуемыми для использования, т.е. элементы, которые являются кандидатами на удаление в будущем.

Устаревшие черты противоречат концепции структурного программирования, затрудняют распараллеливание, являются источником ненадежности, снижают мобильность и эффективность программ.

Отметим еще, что современный стандарт требует, чтобы компилятор сообщал пользователю не только конструкции, которые не отвечают требованиям языка, но также и устаревшие конструкции.

В Фортране 90 имеется список устаревших черт (которые сохранены для совместимости), но список удаленных черт — пустой. В Фортране 95 удалены некоторые черты, которые были признаны устаревшими в предыдущем стандарте. Перечни удаленных и устаревших черт в Фортране 90 и Фортране 95 содержатся в официальных описаниях стандартов [1 – 3]; в работе [26], помимо списков, даются рекомендации по замене устаревших черт современными элементами языка.

14. Перспективы развития языка. В настоящее время опубликован для обсуждения проект будущего стандарта, рабочее название которого Фортран 2003 (на некотором этапе использовались названия Фортран 200x и Фортран 2000). Предполагается, что стандарт Фортран 2003 будет утвержден в конце 2004 года.

Фортран 2003 предусматривает весьма существенные нововведения. Ниже перечислены некоторые из них:

— развитие средств объектно-ориентированного программирования (полный набор средств ООП; вводятся расширяемые типы для реализации наследования и средства реализации динамического полиморфизма — полиморфные объекты, т.е. объекты, тип которых варьируется во время выполнения программы, и др.);

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

— 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 [27];

— новые возможности, касающиеся размещаемых массивов [28];

— более полная интеграция с операционной системой.

Полный текст официального описания проекта стандарта можно найти в Интернете [29].

В работе [30] содержатся некоторые предложения по дальнейшему развитию языка после утверждения Фортрана 2003. Предполагается, что следующий стандарт будет содержать небольшие расширения по сравнению с Фортраном 2003.

Заключение. Современные международные стандарты языка Фортран (Фортран 90 и Фортран 95), как отмечалось в статье, являются результатом совместной деятельности экспертов многих стран. Несмотря на то, что наша страна является официальным членом соответствующих международных структур, в России фактически не поддерживается участие специалистов в деятельности рабочей группы экспертов по Фортрану.

По нашему мнению, было бы весьма полезным более тесная интеграция российских ученых в международную деятельность по развитию и стандартизации Фортрана — языка, который лидирует при решении больших вычислительных задач на современных высокопроизводительных ЭВМ.

Это было бы полезно и для престижа страны, и для того, чтобы в международных стандартах учитывались традиции, опыт отечественных специалистов и особенности задач, решаемых в нашей стране.

Хочу остановиться еще на одной проблеме, имеющей непосредственное отношение к теме, которая обсуждалась в этой статье.

Известные ученые нашей страны неоднократно обращали внимание на то, что в России “индустрия программного обеспечения” занимает слишком скромное место (см., например, [31, 32]), несмотря на то, что в стране имеются все предпосылки для создания и развития этой “экспортной отрасли”. Причины такого положения дел и пути выхода из этой ситуации неоднократно отмечались в печати.

Хотелось бы обратить внимание еще на одну причину, которая в нашей стране не всегда учитывается. Для того чтобы прикладная программа стала программным продуктом и была конкурентноспособной на зарубежном рынке, она должна быть написана на языке, который является международным стандартом, с использованием современных технологий, предоставляемых этим стандартом. К сожалению, этим вопросам не уделяется должного внимания в курсах программирования в высших учебных заведениях нашей страны [33].

Если говорить о Фортране, которому посвящена данная статья, то старые варианты языка, которые еще часто используются в нашей стране, не позволяют применять современные технологии. **Для программирования прикладных задач вычислительного характера, решаемых на современных компьютерах, предпочтение следует отдавать стандартным языкам с использованием при разработке программ современных технологий, которые обеспечивают эти стандарты. Это**

позволит создавать конкурентоспособный программный продукт и, по нашему мнению, позволит выйти на рынок не только с разработанными в нашей стране вычислительными алгоритмами и методиками расчетов (которые высоко ценятся за рубежом), но и с программным продуктом.

СПИСОК ЛИТЕРАТУРЫ

1. ISO/IEC 1539-1: 1997. Information technology–Programming languages–Fortran. Part 1: Base Language.
2. ISO/IEC 1539: 1991(E). Information technology–Programming languages–Fortran.
3. Фортран 90. Международный стандарт / Пер. с англ. С. Г. Дробышевич, редактор перевода А. М. Горелик. М.: Финансы и статистика, 1998.
4. ISO/IEC 1539-2: 2000(E). Information technology–Programming languages–Fortran. Part 2: Varying length strings.
5. ISO/IEC 1539-3: 1998. Information technology–Programming languages–Fortran. Part 3: Conditional compilation.
6. *Adams J., Brainerd W., Martin J., Smith B., and Wagener J.* Fortran 90. Handbook. (Complete ANSI/ISO Reference). New York: McGraw Hill, 1992.
7. *Metcalf M., Reid J.* Fortran 90/95 Explained (2nd edition). Oxford: Oxford University Press, 1999.
8. *Горелик А.М., Ушкова В.Л.* Фортран сегодня и завтра. М.: Наука, 1990.
9. *Меткалф М., Рид Дж.* Описание языка программирования Фортран 90 / Пер. с англ. П. А. Горбунова. М.: Мир, 1995.
10. *Горелик А.М.* Современные международные стандарты языка Фортран // Программирование. 2001. № 6. 44–56 (English translation: Gorelik A.M. Up-to-date international standards of the Fortran programming language // Programming and Computer Software. 2001. **27**, N 6. 320–328).
11. *Горелик А.М.* Фортран жил, жив и будет жить // Открытые системы сегодня. 1995. № 1. 6.
12. *Горелик А.М.* Современный Фортран для компьютеров традиционной архитектуры и для параллельных вычислительных систем. Препринт ИПМ им. М. В. Келдыша РАН. 2003. № 29.
13. http://www.fortran.com/
14. *Горелик А.М.* Средства поддержки мобильности и надежности программ в современном Фортране. Препринт ИПМ им. М. В. Келдыша РАН. 2000. № 55.
15. *Горелик А.М.* Объектно-ориентированное программирование на современном Фортране // Препринт ИПМ им. М. В. Келдыша РАН. 2002. № 70.
16. *Горелик А.М.* Объектно-ориентированное программирование на современном Фортране // Программирование. 2004. № 3 (в печати).
17. *Горелик А.М.* Средства явной спецификации векторных операций и их использование для программирования вычислительных задач. Препринт ИПМ им. М. В. Келдыша РАН. 2003. № 70.
18. *Decyk V., Norton C., and Szymanski Z.* How to express C++ concepts in Fortran 90 // Scientific Programming. 1997. **6**, N 4. 363–390.
19. *Cary J., Shasharina S., Cummings J., Reinders J., and Hinker P.* Comparison of C++ and Fortran 90 for object-oriented scientific programming // Computer Physics Communications. 1997. **105**.
20. *Горелик А.М.* Средства поддержки параллельности в современном Фортране. Препринт ИПМ им. М. В. Келдыша РАН. 1999. № 75.
21. Векторизация программ: теория, методы, реализация: Сб. статей / Пер. с англ. и нем. под ред. Г. Д. Чинина. М.: Мир, 1991.
22. *Luecke G., Haque W., Hoextra J., Jespersen L., and Coyle J.* Evaluation of Fortran vector compilers and preprocessors // Software–Practice and Experience. 1991. **21**, N 9. 891–905.
23. *Ina H., Kamiya S., and Mikami J.* Languages and software development tools for supercomputers // Computer Phisics Communications. 1985. **38**. 211–219.
24. *Задыхайло И.Б., Зеленецкий С.Д., Платонова Л.Н., Поддерюгина Н.В., Седова И.М., Эйсымонт Л.К.* ФОРА–ЕС: Система программирования Фортран IV для многопроцессорного вычислительного комплекса ПС-3000. Препринт ИПМ им. М. В. Келдыша АН СССР. 1987. № 17.
25. *Платонова Л.Н., Горелик А.М., Задыхайло И.Б., Зеленецкий С.Д., Поддерюгина Н.В.* Расширение языка Фортран для супер-ЭВМ // Проблемы повышения эффективности использования ЭВМ большой производительности. М.: ВЦ АН СССР, 1989.
26. *Горелик А.М.* Эволюция языка Фортран. Устаревшие черты и современные элементы языка для их замены. Препринт ИПМ им. М. В. Келдыша РАН. 1997. № 66.
27. Floating Point Exception Handling. ISO/IEC TR 15580: 2001.
28. Enhanced Data Type Facilities. ISO/IEC TR 15581: 1998.
29. htpp://www.j3-fortran.org/
30. *Nagle D.* Next Standard. ISO/IEC JTC1/WG5 № 1460.
31. *Фортов В.Е.* Индустрия программного обеспечения — это шанс для России // Известия. 2000. 22 ноября.
32. *Фортов В.Е.* Обустроить в России Силиконовую долину // Известия. 2002. 15 марта.
33. *Горелик А.М.* О целесообразности изучения современного Фортрана в вузах // Программирование. 1996. № 3. 79–80 (English translation: Gorelik A.M. On the expedience of studying modern Fortran in universities // Programming and Computer Software. 1996. **22**, N 3. 162).