ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В. КЕЛДЫША
РОССИЙСКОЙ АКАДЕМИИ НАУК

А.М.Горелик

СРЕДСТВА ПОДДЕРЖКИ МОБИЛЬНОСТИ И НАДЕЖНОСТИ ПРОГРАММ В СОВРЕМЕННОМ ФОРТРАНЕ

Москва
2000

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 00-01-00043 и 00-01-298).

Средства поддержки мобильности и надежности программ
в современном Фортране

А.М.Горелик

АННОТАЦИЯ

В работе анализируются новые средства современных стандартов языка Фортран, которые позволяют создавать более мобильные и более надежные прикладные программы по сравнению с предшествующими стандартами языка. Обсуждение ориентировано на программирование для традиционной архитектуры и для параллельных компьютеров. Приводятся некоторые предложения по усовершенствованию языковых средств и даются рекомендации по написанию мобильных и надежных программ.

This work was supported by Russian Foundation for Fundamental Investigations (Projects 00-01-00043 and 00-01-00298).

Support of Portability and Reliability in Modern Fortran

A.M.Gorelik

The preprint of the Keldysh Institute of Applied Mathematics,
Russian Academy of Sciences

ABSTRACT

The following aspects of modern Fortran Standards are analysed: new features which promote to elaborate more portable and more reliable applications as compared with previous language standards. Analysis is oriented both to traditional architectures and to parallel computers. Some recommendations for users and some suggestions are given.

Содержание

1. Введение

Язык Фортран остается широко распространенным языком программирования при решении научно-технических задач, требующих большого объема вычислений. Разумеется, речь идет о современном Фортране, а не о Фортране 77, который, естественно, устарел и не соответствует требованиям времени. Интерес к Фортрану возродился после вступления в действие современных международных стандартов языка [1-5] и реализации их практически для всех вычислительных систем, а также в связи с использованием высокопроизводительных параллельных компьютеров.

Анализу различных аспектов современных стандартов языка Фортран посвящены работы [6-9]. В данной работе анализируются другие аспекты этих стандартов, а именно, средства поддержки мобильности и надежности создаваемых программ. Рассматриваются такие новые средства, которые обеспечивают возможность создания более мобильных и более надежных прикладных программ по сравнению с предшествующими, но еще используемыми в нашей стране версиями языка; кроме того, указываются те средства, которые недостаточны или препятствуют достижению высокой мобильности или надежности программ и приводятся некоторые рекомендации для прикладных программистов.

Рассматриваются не только средства, ориентированные на традиционную архитектуру, но также средства поддержки мобильности и надежности, ориентированные на параллельную обработку.

2. Понятие мобильности программ

Мобильность - это свойство программы, выражающееся в возможности ее адаптации для работы в различных окружениях. Можно говорить о высокой или низкой мобильности программ, понимая при этом большую или меньшую степень легкости ее адаптации к вычислительной среде.

Высокая мобильность облегчает использование ранее созданного прикладного программного обеспечения и перенос программы с одной платформы на другую, что приводит к существенному уменьшению времени разработки и стоимости программного обеспечения.

Мобильность программного обеспечения зависит от многих факторов. Отметим некоторые из них.

Мобильность в значительной степени зависит от распространенности используемого языка программирования, его стандартизации, точности и строгости эталонного описания, а также от имеющихся в языке специальных средств написания мобильных программ. Мобильность программ косвенно зависит от

богатства выразительных средств языка. Мобильность программ во многом определяется также дисциплиной и технологией программирования, реализацией языка, качеством документации и другими факторами.

Следует отметить, что достижение высокой мобильности программ может потребовать некоторого усложнения соответствующих алгоритмов. Степень усложнения тем меньше, чем лучше язык оснащен средствами поддержки мобильности.

Мобильность программного обеспечения зависит и от ряда других факторов. Более подробно эти проблемы анализируются в работах [11-13].

В данной работе мы ограничимся обсуждением мобильности, которая обусловлена только свойствами языка; рассматриваются такие новые возможности в современных стандартах Фортрана, которые облегчают адаптацию программ к конкретному окружению или конкретным условиям функционирования, т.е. являются предпосылкой более высокой мобильности по сравнению с Фортраном 77.

Обсуждаемые ниже аспекты мобильности (разделы 3-7) имеют отношение к компьютерам любой архитектуры. Проблемам мобильности параллельных систем посвящен специальный раздел 6.7.

3. Стандартизация языков программирования – основа повышения мобильности программ

Стандартизация языков программирования создает предпосылки для повышения мобильности программного обеспечения. Использование международных стандартов языков программирования является одним из наиболее важных условий разработки мобильных программ для компьютеров любой архитектуры.

Вопросам стандартизации языков программирования, в т.ч. и Фортрана, во многих странах уделяется большое внимание, в этой деятельности участвуют многие ведущие фирмы, научные и учебные центры.

К сожалению, в нашей стране роль стандартов на языки программирования явно недооценивается. Высказывается иногда мнение, что стандартизация языка тормозит его развитие. Однако это утверждение ошибочно, так как в ISO (Международная организация по стандартизации) предусмотрен механизм периодического пересмотра стандартов.

Кроме того, все стандарты разрешают включать в реализацию языка дополнительные средства, не предусмотренные стандартом. При этом современные стандарты Фортрана требуют, чтобы компиляторы выдавали сообщения о

несоответствии стандарту, об используемых расширениях. Программист, использующий дополнительные средства, должен понимать, что они могут отсутствовать при переносе его программы в другую вычислительную среду.

Многие разработчики программного обеспечения четко выделяют такие расширения в документации. Однако в литературе, содержащей описание конкретных реализаций, к сожалению, не всегда четко разграничивается, какие черты являются нестандартными, что затрудняет разработку мобильных программ.

Еще больше проблем для создания мобильных программ представляют реализации, которые не соответствуют стандарту. Если программа ориентирована на такую реализацию, то затраты на ее переделку для использования в другой вычислительной среде в некоторых случаях могут быть сопоставимы с разработкой новой программы.

Учитывая сказанное, еще раз отметим, что главным условием написания мобильных программ является использование международных стандартов (см. также раздел 7, п.1).

4. Современные международные стандарты Фортрана

Современные стандарты языка Фортран позволяют создавать более мобильные программы по сравнению с Фортраном 77, который пока еще используется в нашей стране.

Прежде чем рассматривать конкретные средства поддержки мобильности в современных стандартах Фортрана приведем краткие сведения об этих стандартах.

В 1991 году утвержден международный стандарт языка Фортран, получивший неформальное название Фортран 90 [1-4]. Фортран 90 обеспечивает современный стиль программирования, он существенно расширяет возможности своего предшественника Фортрана 77, одновременно практически сохраняет полную с ним преемственность.

Следующим этапом в развитии современного Фортрана было принятие в 1997 году нового стандарта, неформальное название которого - Фортран 95 [5]. По сравнению с Фортраном 90 в Фортране 95 новшеств сравнительно немного.

Современные стандарты Фортрана представляют собой семейство стандартов, состоящее из нескольких частей. Действующий стандарт Фортран 95 состоит из трех частей.

Первая часть - основной (базовый) язык. Остальные части являются дополнительными. При этом не требуется, чтобы компилятор, соответствующий стандарту, обязательно реализовывал дополнительные части.

Вторая часть стандарта содержит описание средств для работы с символьными строками переменной длины. Третья часть Фортрана 95 определяет описание языка условной компиляции.

Развитие Фортрана продолжается. В настоящее время разрабатывается проект будущего стандарта, рабочее название которого Фортран 2000 или Фортран 200x [8,10] (на последней встрече рабочей группы в Финляндии в августе 2000 г. предпочтение было отдано названию Фортран 2000).

Для того чтобы эффективно реагировать на новые требования, не дожидаясь принятия следующего стандарта (Фортран 2000), было решено выделить наиболее приоритетные новые черты и выработать проекты для таких новых черт в качестве технических отчетов. В дальнейшем такие средства должны быть вставлены в неизмененном виде в следующую ревизию основного стандарта (Фортран 2000), кроме, возможно, необходимых модификаций, в случае, если опыт реализации и использования покажет, что требуются существенные изменения (окажутся серьезные ошибки в описании или встретятся сложности при интеграции с другими новыми возможностями).

Эти технические отчеты позволяют реализаторам добавить новые черты в компиляторы Фортрана 95, не дожидаясь завершения разработки будущего стандарта в полном объеме. Если потребуются какие-либо модификации языка, то разработчики будут стремиться к тому, чтобы минимизировать изменения в существующих коммерческих реализациях.

К настоящему времени завершена разработка следующих двух технических отчетов, которые одобрены и уже являются стандартами:

- Новые возможности, касающиеся размещаемых массивов.
- Исключительные ситуации для операций с плавающей точкой.

Проблемы стандартизации средств параллельной обработки рассматриваются ниже в 6.7.

5. Факторы, затрудняющие адаптацию программ к вычислительной среде

Причины, затрудняющие перенос программы в другую вычислительную среду и адаптацию ее к новым условиям, обусловлены следующими различиями в конкретных реализациях:

- расширения стандарта, которые имеются не во всех реализованных версиях;
- различная степень контроля запретов и ограничений стандарта;

- различная трактовка неясных мест стандарта;
- различия в технических ограничениях (глубина вложенности, число параметров процедур и т.п.);
- различные способы представления данных;
- набор выделенных номеров для внешних устройств (экран, клавиатура, принтер и т.п.) и способ формирования имен файлов;
- использование процедур, которые отсутствуют в стандарте;
- разнообразие архитектур, ориентированных на параллельную обработку,

и др.

6. Новшества в современных стандартах Фортрана, способствующие повышению мобильности создаваемых программ

Наиболее существенными факторами, способствующими повышению мобильности программ, являются следующие: богатство выразительных средств языка и наличие в языке мобильных средств адаптации к конкретному окружению. Из средств адаптации выделим средства адаптации по точности и диапазону представления данных, средства задания параметров окружения и запросов к обстановке, средства параметризации программ, средства обработки исключительных ситуаций и др. Ниже эти факторы рассматриваются более подробно.

6.1. Богатство выразительных средств языка как условие повышения мобильности

Под выразительными средствами языка обычно понимают совокупность имеющихся в нем элементов для отображения данных и алгоритмов их обработки. Другими словами, к выразительным средствам относятся разнообразие и гибкость типов и видов данных, полнота набора операций и способов конструирования выражений, состав управляющих конструкций, возможность создавать новые типы данных и новые операции, средства инкапсуляции и наследования, удобные средства использования процедур, взаимодействие с процедурами, написанными на других языках, удобство средств ввода-вывода, наличие средств управления памятью и др.

Богатство выразительных средств, безусловно, способствует повышению мобильности программ, хотя и несколько косвенным образом.

Во-первых, разработчики реализаций языка, богатого выразительными средствами, имеют меньше поводов для внесения в язык собственных расширений,

которые часто не совместимы с другими реализациями. Во-вторых, даже если конкретная версия языка и содержит расширения по сравнению с эталонной версией, у программистов будет меньше соблазнов воспользоваться расширениями в своих программах.

Современные стандарты Фортрана, в отличие от предшественников, обладают богатым набором выразительных средств. Наиболее важные новые возможности - следующие: свободный формат исходной программы, производные типы данных (структуры данных), средства параметризации встроенных типов данных, операции над массивами и секциями массивов как над целыми объектами, указатели, механизмы динамического размещения объектов в памяти, новые управляющие конструкции, определяемые пользователем операции, модули, элементы объектно-ориентированного программирования, более развитые средства использования процедур и др.

6.2. Адаптация по представлению данных

В программах вычислительного характера важным фактором повышения мобильности является возможность адаптации по точности и диапазону данных. Это объясняется тем, что использование программы на компьютере с формой представления данных отличной от той, для которой первоначально разрабатывалась программа, может привести к тому, что будут возникать ситуации "переполнения" или "исчезновения порядка".

В Фортране традиционно существовало два встроенных типа вещественных данных: REAL и DOUBLE PRECISION; эти типы охватывают весь диапазон значений вещественных чисел, представимых в машинном слове; точность представления этих данных зависит от ЭВМ и реализации.

Наличие двух встроенных типов вещественных данных позволяет в случае необходимости (например, при переходе на другую ЭВМ с меньшей разрядностью машинного слова) так модифицировать программу, чтобы в ней использовались типы, обеспечивающие нужные характеристики представления данных. Однако такая модификация может оказаться довольно трудоемкой. Кроме того, встроенные типы позволяют лишь косвенно регулировать точность и диапазон, так как задают относительную длину представления данных в реализации.

В современных стандартах введены и более совершенные средства задания характеристик представления вещественных данных. В Фортране 90/95 для объекта любого встроенного типа, в том числе вещественного, может быть задан параметр

типа - способ представления, т.е. каждый встроенный тип может иметь несколько разновидностей.

Так, для вещественного типа параметр можно выбрать так, чтобы обеспечить требуемые значения точности и диапазона.

Пример

```
! Выбор способа представления, обеспечивающего точность
! не менее 8 значащих цифр и диапазон десятичного порядка
!   не менее чем  (-70, +70)
INTEGER, PARAMETER :: NUMS = SELECTED_REAL_KIND (8, 70)
! NUMS - параметр типа
REAL (KIND = NUMS) R1, R2
```

Для приведенного примера значения переменных R1 и R2 на одних компьютерах могут отображаться как обычные вещественные числа, занимающие одно машинное слово, на других - как "длинные" (двойной точности) вещественные числа; однако при переносе данной программы на компьютер с иной формой представления данных не требуется менять ни описания переменных R1 и R2, ни другие операторы, связанные с этими переменными, так как в любой реализации, если компилятор не выдал отказа, должны быть обеспечены соответствующие точность и диапазон. Таким образом, такие средства больше отвечают требованиям мобильности программ, чем использование традиционных непараметризованных типов REAL и DOUBLE PRECISION.

Средства параметризации имеются и для других встроенных типов. Ниже приведены примеры описания переменных целого типа с заданным параметром типа, который специфицирует диапазон.

Примеры

```
INTEGER (2) I2
INTEGER, PARAMETER :: K = SELECTED_INT_KIND(6)
INTEGER (K) J1, J2
```

Отсутствие в прежних стандартах Фортрана средств задания точности и диапазона представления данных, как показала практика, отрицательно сказывается на мобильности. Действительно, во многих реализациях Фортрана, появились не предусмотренные стандартом типы REAL*4, INTEGER*2 и др., что снижает мобильность программ. С учетом этих факторов введение в язык параметров встроенных типов является средством повышения мобильности.

6.3. Параметры окружения

Имеющиеся в языках параметры окружения обычно служат для адаптации программы к конкретным аппаратным средствам, что является предпосылкой для написания мобильных программ.

В Фортране 90/95 для таких целей служат числовые справочные функции: DIGITS(X), EPSILON(X), HUGE(X), MAXEXPONENT(X), MINEXPONENT(X), PRECISION(X), RADIX(X), RANGE(X), TINY(X).

Эти функции описываются в терминах модели представления чисел в процессоре. Модель имеет параметры, которые определяются так, чтобы наилучшим образом приспособиться для машины, на которой будет выполняться программа. Функции выдают различные характеристики моделей, связанных с их аргументами. Использование значений этих характеристик для данных целого и вещественного типов для каждой из реализованных разновидностей этих типов позволяет настроить программу к конкретному окружению. Более подробно (см. в [1, 4, 5]).

6.4. Параметризация программ. Именованные константы

Одним из средств повышения мобильности программ является их параметризация. Параметризация заключается в том, что в программе выделяются параметры настройки, которые обеспечивают настройку в конкретное окружение в соответствии с параметрами данного окружения.

Присваивание значений таким параметрам (если оно необходимо) выносится в начало программы, и при адаптации программы в конкретное окружение или при другой перенастройке программы коррекция значений этих параметров будет не слишком трудоемкой.

Наиболее удобным средством для отображения параметров настройки программ в Фортране являются именованные константы, которые задаются оператором PARAMETER (или с помощью атрибута PARAMETER в операторах объявления типа Фортрана 90/95).

Примеры использования таких параметров для задания параметров типа приведены выше в 6.2.

Другим примером может служить использование параметров для спецификации устройств ввода/вывода. Для написания мобильных программ в операторах ввода/вывода не рекомендуется указывать явно в виде литеральных констант номера устройств (клавиатуры, экрана, принтера и т.п.), лучше использовать именованные константы. Это объясняется тем, что в стандарте за

устройствами ввода/вывода не закреплены фиксированные номера, а в конкретных реализациях такое закрепление имеет место.

Еще один пример: границы статических массивов целесообразнее специфицировать с помощью именованных констант; в случае изменения границ потребуется изменить лишь значения соответствующих констант.

Пример

```
INTEGER, PARAMETER :: M = 10, N=20
REAL A(M, N)
```

Новым в современных стандартах является возможность задания именованных констант в программной единице MODULE. Это позволяет устанавливать значения параметров только в одной программной единице-модуле, а не в каждой программной единице, где это требуется.

Другим новшеством является возможность задания параметра более современным способом, чем в операторе PARAMETER, - с помощью атрибута PARAMETER в операторе объявления типа (см. приведенный выше пример).

6.5. Порядок вычислений и побочный эффект

В стандарте не фиксируется последовательность выполнения операций при условии соблюдения старшинства операций и целостности выражений внутри скобок (см.[1, 5], 7.1.7.3.).

Между тем, вследствие приближенного способа представления вещественных чисел и операций над ними в ЭВМ, результат (или точность результата) может зависеть от порядка вычислений. Порядок вычислений устанавливается компилятором в соответствии с принятой в нем схемой реализации и может быть различным даже для одного компилятора (например, может зависеть от уровня оптимизации). Таким образом, нестрогий порядок вычисления выражений может стать источником недостаточной мобильности и низкой надежности программ. Если порядок вычисления выражений является существенным, программисту, особенно при написании мобильных программ, рекомендуется использовать скобки, указывающие последовательность выполнения операций.

При использовании в выражениях функций с побочным эффектом результат может зависеть от реализации, что также является проблемой для написания мобильных программ. Для преодоления этой неопределенности в Фортране 95 для описания функций без побочного эффекта введен атрибут PURE. Это позволяет компилятору реализовать вычисление выражения, содержащего вызов такой функции, без снижения мобильности.

6.6. Средства условной компиляции

Средства условной компиляции также являются хорошим инструментом для повышения мобильности программ в тех случаях, когда пользователю необходимо поддерживать более одной версии исходной программы.

Поддерживать несколько версий, если программа большая, иногда затруднительно. Для программиста было бы желательно сохранить один исходный текст, который содержит внутри себя все варианты, так, чтобы любая версия могла быть легко извлечена. Это обычно решается следующим образом: в исходный текст вставляются дополнительные строки для управления процессом компиляции, и все строки исходного текста, которые не выбраны, опускаются или заменяются комментариями, а те, которые выбираются, передаются компилятору.

До недавнего времени средства условной компиляции отсутствовали в стандартах Фортрана. В настоящее время они включены в третью дополнительную часть стандарта (см. выше в разделе 4). Более подробно набор таких средств описан в [8].

6.7. Мобильность для параллельных программ

Одной из наиболее актуальных проблем в развитии Фортрана является разработка языковых средств, ориентированных на параллельную обработку.

Разработаны и продолжают разрабатываться как специальные языки для параллельного программирования, так и расширения для традиционных языков, включая, естественно, и Фортран.

Очевидно, что для обеспечения мобильности параллельных программ необходима стандартизация и унификация таких средств. Унификация средств распараллеливания в языках высокого уровня - довольно сложная проблема. Трудность заключается в том, что имеется большое разнообразие архитектурных решений, позволяющих использовать возможности распараллеливания. Тем не менее, в настоящее время уже разработаны системы, включающие средства, которые стали фактическими стандартами для параллельных вычислительных систем различной архитектуры, хотя не все пока имеют статус официальных международных стандартов. Эти средства реализованы для многих параллельных компьютеров. Ниже приводятся краткие сведения об этих средствах.

В Фортране 90/95 имеется большой набор средств для работы с массивами и секциями массивов как с целыми объектами. Эти средства неявно специфицируют параллелизм действий над компонентами массива или нескольких массивов. Они позволяют в лаконичной и сжатой форме описать алгоритмы обработки массивов и

позволяют компилятору сгенерировать эффективный код с учетом особенностей аппаратуры.

При написании параллельных программ серьезной проблемой является такое свойство программы как детерминизм. Даже для одной и той же вычислительной среды возможны ситуации, в которых одна и та же программа может вырабатывать разные результаты. Некоторые средства предотвращения недетерминизма обычно предусматриваются в системах, ориентированных на параллельную обработку. Тем не менее, в ряде случаев ответственность по обеспечению приемлемого функционирования программы лежит на программисте.

Из мер, которые введены в современные стандарты, отметим атрибут PURE, который используется для спецификации функций без побочного эффекта. Вызов такой функции можно использовать в тех случаях, где возможна параллельная обработка, без таких нежелательных последствий как недетерминизм.

Хотя стандарты Фортран 90 и Фортран 95 содержат некоторые средства поддержки параллельности, они не являются полностью параллельными языками. Однако разработанные на базе этих стандартов параллельные языки существенно используют новые возможности Фортрана 90/95 (динамические и размещаемые массивы, указатели, структуры, модули, явный интерфейс, определяемые пользователем операции, большое число новых встроенных процедур и др.).

Для систем с распределенной памятью фактическим стандартом является система MPI (Message Passing Interface), которая представляет собой набор библиотечных интерфейсов. Система использует механизм задач (процессов) и передачи сообщений (обмен данными). Исходная программа разбивается на задачи (подзадачи), которые могут выполняться параллельно на разных процессорных узлах, и связь между задачами осуществляется с помощью передачи сообщений. Относительно низкий уровень библиотечных средств, чрезмерная детализация приводят к определенным неудобствам при использовании таких систем прикладными программистами, однако, подобные системы являются хорошим инструментальным средством для реализации языков более высокого уровня.

Для вычислительных машин с общей памятью, работы по унификации расширений языков средствами спецификации параллелизма выполнялись первоначально группой PCF (Parallel Computing Forum), в дальнейшем X3H5 ANSI. Проект X3H5 положен в основу стандарта Open MP, который был утвержден в 1997 году (хотя не в рамках ANSI или ISO).

При использовании подхода, ориентированного на разбиение данных, фактическим стандартом является язык HPF (High Performance Fortran) и его более новая версия HPF 2.0. HPF и HPF 2.0 являются расширениям соответственно Фортрана 90 и Фортрана 95.

HPF и HPF 2.0 содержат директивы для описания способов разбиения и распределения данных между параллельно работающими процессорами и средства явного указания параллельности. По утверждению авторов проекта предлагаемые средства могут быть использованы для параллельных компьютеров различной архитектуры.

Директивы HPF имеют вид комментария, который начинается с символов !HPF$. Это позволяет использовать программы с такими комментариями для любых вычислительных систем (включая и последовательные) без снижения мобильности, поскольку компилятор, ориентированный на последовательное выполнение, будет игнорировать такие комментарии. Для того чтобы обеспечить возможность настраиваться на различную архитектуру используются EXTRINSIC процедуры.

Завершая данный раздел, еще раз отметим, что главным условием для достижения мобильности параллельных программ является использование тех средств, которые являются фактическими стандартами.

6.8. Устаревшие черты и мобильность

Некоторые конструкции языка Фортран в настоящее время устарели и стали излишними после введения новых элементов; такие конструкции отнесены к категории устаревших черт. Исключение из языка каких-либо элементов может быть весьма болезненным, так как оно может привести к тому, что невозможно будет использовать существующий фонд программного обеспечения на Фортране, т.е. приведет к снижению мобильности.

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

Учитывая эти противоречивые факторы, было принято следующее решение: элементы языка, которые являются кандидатами на удаление, заранее объявляются устаревшими, нерекомендуемыми для использования; в каждом стандарте языка Фортран должно быть два списка устаревших черт с различным статусом. Первый список содержит удаляемые из данного стандарта черты - те элементы языка, которые в предыдущем стандарте были объявлены как устаревшие. Второй список содержит отживающие (нерекомендуемые) черты - те элементы языка, которые

являются кандидатами на удаление в следующей ревизии стандарта (хотя они необязательно должны быть удалены).

Полный перечень устаревших черт и рекомендации по их замене приведены в работе [6].

При написании мобильных программ программисту не следует использовать устаревшие черты; это позволит облегчить процесс перехода на современные средства.

7. Рекомендации по написанию мобильных программ

Подведем некоторые итоги.

1. Главное требование при написании мобильных программ - строгое соблюдение стандарта. Это касается и программ, ориентированных на параллельную обработку.

В некоторых случаях стандарт не предписывает результаты выполнения программы, когда правила языка не выполняются, и конкретизация неопределенностей возлагается на разработчиков компиляторов. Программисты должны иметь представление о вытекающих отсюда проблемах и избегать ситуаций, которые четко не определены в языке. В большинстве случаев можно обойти эти трудности, запрограммировав соответствующее место другим способом. Однако программисты иногда прибегают к всевозможным "трюкам" и ухищрениям, используя особенности реализации конкретного компилятора, что, естественно, снижает мобильность программы.

В то же время было бы неверным рекомендовать пользоваться в программах только средствами стандарта, особенно в тех случаях, когда программу не предполагается использовать в другой среде или когда используемые расширения имеются во всех тех окружениях, где будет выполняться данная программа. Однако программист, создающий мобильную программу, должен знать стандарт и по возможности его придерживаться. В частности, если в конкретной реализации допускаются разные способы написания некоторой конструкции, рекомендуется использовать средства стандарта. Программист, использующий средства, которых нет в стандарте, должен понимать, что они могут отсутствовать при переносе его программы в другую вычислительную среду.

2. При написании мобильных программ следует выделить элементы зависимости алгоритма от компьютера и операционной системы (значения употребляемых констант, зависящих от среды, требуемую точность вычисления и диапазон данных, номера устройств ввода/вывода и др.).

3. В тех случаях, когда это необходимо, в операторах объявления типов следует использовать параметры типа (см. 6.3.).

4. Рекомендуется использовать имеющиеся в языке средства параметризации программ (см. 6.4.).

5. Если порядок вычисления выражений является существенным, программисту рекомендуется использовать скобки, указывающие последовательность выполнения операций (см. 6.5.).

6. Рекомендуется использовать средства условной компиляции (если в компиляторе они реализованы). Эти средства недавно утверждены в качестве третьей части стандарта Фортрана (см. 6.6).

7. Программисты иногда ошибочно используют передачу управления внутрь цикла или в области цикла помещают операторы, изменяющие значения параметров цикла или управляющей переменной цикла. Такие ошибки не всегда диагностируются и могут привести к разным результатам в разных реализациях.

8. При написании мобильных программ программист не должен использовать устаревшие черты (см.6.8.). В частности, не рекомендуется использовать операторы GO TO со списком (GO TO по предписанию и GO TO по назначению).

9. Рекомендуется отмечать те места программы, где используются непараметризованные величины, зависящие от реализации, и/или расширения конкретной реализации. Это можно сделать с помощью специальных меток, либо с помощью комментариев или указать в документации. Это обелегчит поиск таких мест, если понадобится адаптировать программу к новой вычислительной среде.

8. Понятие надежности

В разделах 8-10 анализируются те черты современных стандартов Фортрана, которые способствуют или, наоборот, препятствуют написанию надежных программ.

Под надежностью здесь понимается независимость программы от частных особенностей реализации и ее устойчивость по отношению к изменению условий функционирования. Мобильность и надежность программ связаны между собой. Программы, в которых некорректно использованы какие-либо особенности реализации, могут "правильно" работать в одной вычислительной среде, но при переносе в другую вычислительную среду оказаться неправильными. Некоторые примеры подобных ситуаций для старого Фортрана приводятся в книге [11].

Структура языка, набор его основных понятий и конструкций влияют на стиль программирования, на ясность, простоту и, как следствие, на надежность программ.

Надежность программ существенно зависит также от полноты диагностики компилятора и его исполнительной системы (что, в свою очередь, в определенной степени зависит от свойств языка программирования).

Достижение высокой надежности программ может потребовать от программиста более осторожного использования отдельных конструкций и принятия некоторых специальных мер. Чем лучше язык оснащен средствами поддержки надежности, тем меньше программисту надо специально заботиться о надежности программ.

Далее рассматриваются некоторые новые языковые элементы, введенные в современные стандарты, которые влияют на надежность. Наиболее важные из них следующие: средства, обеспечивающие возможность контроля типов на этапе компиляции, средства спецификации назначения формальных параметров, средства реакции на особые ситуации, возможность зависимой сепаратной компиляции и др.

9. Новшества в современных стандартах Фортрана, позволяющие повысить надежность создаваемых программ

9.1. Обязательность объявления типов данных

Для языка Фортран традиционно характерен принцип умолчания, т.е. автоматическое приписывание объектам типа, если тип объекта не указан явно. Приписывание типа по умолчанию, а также возможность использования имен, совпадающих со служебными словами, плохо согласуется с требованием надежности.

Для повышения надежности в Фортране 90/95 введен оператор IMPLICIT NONE. При наличии такого оператора типы объектов в данной программной единице должны быть объявлены явно. Явное описание объектов обеспечивает возможность контроля типов, что, очевидно, полезно для надежности.

IMPLICIT NONE является более гибким инструментом, чем просто обязательность описаний, что имеет место в других языках.

9.2. Способы спецификации параметров процедур, повышающие надежность

Полезной для надежности является предусмотренная в Фортране 90/95 возможность спецификации назначения формальных параметров - как входных, выходных или универсальных INTENT(IN), INTENT(OUT), INTENT(INOUT).

Использование этой спецификации позволяет на этапе компиляции обнаруживать некоторые виды ошибок. Так, все параметры, объявленные как

входные, можно защитить от изменения в теле процедуры. Спецификация назначения формальных параметров в Фортране 90/95 может, например, обезопасить программы от ошибки, называемой условно "порчей" констант, связанной с некорректным использованием параметров процедур; эта ошибка не могла быть выявлена в предыдущих вариантах Фортрана.

Другой полезной для надежности чертой в современном Фортране является возможность задания фактических параметров с ключевыми словами по именам формальных параметров. Это средство не только удобно, но также позволяет создавать более надежные программы, чем при использовании позиционных параметров.

Важной чертой обеспечения надежности является введение в современные стандарты механизма, который обеспечивает явный интерфейс при вызове процедур. В Фортране 77 интерфейс с вызываемой процедурой был только неявным, т.е. при обращении к процедуре отсутствует или является неполной информация о формальных параметрах вызываемой процедуры. Поскольку программные единицы, содержащие внешние процедуры, обычно компилируются раздельно и независимо, неявный интерфейс, очевидно, не позволяет контролировать правильность обращений к процедурам на этапе компиляции. Это обстоятельство является одной из причин недостаточной надежности программ, которую нельзя устранить в предыдущих вариантах Фортрана.

Явный интерфейс, который обеспечивается с помощью введенного в современные стандарты Фортрана интерфейсного блока, означает, что на этапе компиляции процессору (компилятору) доступна вся информация о формальных параметрах вызываемой процедуры и атрибутах результатов функции. Тем самым явный интерфейс позволяет компилятору организовать проверку правильности обращения к процедуре уже на этапе компиляции, что повышает надежность создаваемых программ.

9.3. Зависимая раздельная компиляция

Раздельная компиляция, традиционно предусматриваемая и широко используемая в Фортране 66 и Фортране 77, является независимой - не существует средств контроля связей программных единиц (единиц компиляции). В Фортране 90 в этом направлении сделан шаг вперед: появились уже упоминавшиеся языковые средства для спецификации явного интерфейса процедур - интерфейсные блоки, а также программные единицы-модули и средства доступа к глобальным объектам модулей. Тем самым появилась возможность организовать раздельную зависимую

компиляцию программных единиц, т.е. компиляция может осуществляться с полным контролем связей единиц компиляции что, безусловно, повышает надежность программ.

9.4. Средства реакции на особые ситуации

Важным фактором повышения надежности является возможность задания предопределенных в языке и определяемых программистом особых, или исключительных ситуаций, в том числе ситуаций, связанных с ошибками арифметических операций, ввода/вывода и др.

В языках Фортран 77, Фортран 90/95 имеется возможность задать действия в случае особых ситуаций при выполнении операторов ввода/вывода (спецификации IOSTAT, ERR и END). Новым является средство, позволяющее в Фортране 90/95 задать спецификацию ошибки в операторах размещения и удаления массивов (спецификация STAT в операторах ALLOCATE и DEALLOCATE).

Как отмечалось выше, к настоящему времени завершена разработка технического отчета "Исключительные ситуации для операций с плавающей точкой", который имеет статус стандарта.

9.5. Новые средства повышения надежности в операторах ввода/вывода

Введение в Фортран 90/95 спецификации ACTION(READ), ACTION(WRITE), ACTION(READWRITE) (чтение, запись или чтение/запись) в операторах открытия файлов, очевидно, способствует повышению надежности, так как предохраняет файл от несанкционированного доступа.

9.6. Надежность параллельных программ

Проблема детерминизма, о которой упоминалось в 6.7, является важной не только для мобильности, но также и для надежности программ. Использование атрибута PURE для функций без побочного эффекта является одной из мер предотвращения недетерминизма.

10. Средства, препятствующие надежности

Серьезным препятствием для контроля правильности использования типов является употребление операторов EQUIVALENCE. Действительно, использование общей памяти для нескольких переменных или массивов приводит к тому, что изменение одного объекта может повлечь за собой автоматическое изменение другого, что проконтролировать не всегда возможно.

В связи с этим не рекомендуется использовать оператор EQUIVALENCE, вместо него лучше использовать более современные средства. Оператор EQUIVALENCE обеспечивает совместное использование памяти несколькими объектами данных; такая возможность используется обычно либо для экономии памяти (что было важно, когда размер памяти был недостаточным), либо для моделирования структуры данных с компонентами разных типов. Однако, во-первых, этот искусственный прием не удобен, и, во-вторых, оператор EQUIVALENCE имеет ряд ограничений.

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

11. Рекомендации по написанию надежных программ

Подводя итоги вышеизложенному, приведем некоторые рекомендации по повышению надежности создаваемых программ.

1. Желательно использовать оператор IMPLICIT NONE (см. 9.1.) и явное объявление типов всех используемых в программе объектов.

2. При спецификации внешних процедур рекомендуется использовать атрибут INTENT (см. 9.2.).

3. При вызове процедур полезно использовать ключевые параметры и явный интерфейс (см. 9.2. и 9.3.).

4. Рекомендуется использовать имеющиеся средства реакции на особые ситуации (см. 9.4.).

5. При открытии файла полезно использовать спецификацию ACTION (см. 9.6.).

6. Не следует использовать оператор EQUIVALENCE (см. 10.).

12. Некоторые предложения по внесению в язык средств для повышения мобильности и надежности

1. Для повышения мобильности было бы полезным ввести в стандарт предопределенные константы-параметры окружения.

2. Ввести в стандарт средства, обеспечивающие удобный интерфейс с Си-функциями, также весьма полезно, в том числе и для мобильности.

В программах, написанных на Фортране, часто возникает необходимость использовать процедуры, написанные на Си. Обеспечить удобный способ вызова из Фортран-программ функций, написанных на Си, с помощью имеющихся в Фортране

средств не всегда возможно. В некоторых реализациях вводятся дополнительные средства, облегчающие такой вызов, но эти средства не являются стандартными. В настоящее время разрабатываются такие средства для включения в будущий стандарт, но пока они не введены; использование средств взаимодействия с Си, имеющихся в конкретных реализациях, снижает мобильность программы.

3. Снятие некоторых ограничений, т.е. средств, для которых результат не определен, - полезно для повышения мобильности.

4. Надежности, как уже отмечалось, препятствует использование оператора EQUIVALENCE. Считаем целесообразным, включить этот оператор в список устаревших черт языка с тем, чтобы в дальнейшем исключить его из стандарта.

Литература

1. ISO/IEC 1539: 1991(E) Information technology - Programming languages - Fortran
2. Фортран 90. Международный стандарт. Перевод с англ.С.Г.Дробышевич, редактор перевода А.М.Горелик. М.: Финансы и статистика, 1998
3. Горелик А.М., Ушкова В.Л. Фортран сегодня и завтра. М.: Наука, 1990
4. Меткалф М., Рид Дж. Описание языка программирования Фортран 90.Перевод с английского. М.: Мир, 1995
5. ISO/IEC 1539-1: 1997 Information technology - Programming languages - Fortran - Part 1: Base Language
6. Горелик А.М. Эволюция языка Фортран. Устаревшие черты и современные элементы языка для их замены. - М., Препринт Института прикладной математики РАН, 1997, N66
7. Горелик А.М. Современный Фортран. Состояние и перспективы развития. - М., Препринт Института прикладной математики РАН, 1998, N71
8. Горелик А.М. На пути к Фортрану 200x. Препринт Института прикладной математики им. М.В.Келдыша РАН, 1999, N64
9. Горелик А.М. Средства поддержки параллельности в современном Фортране Препринт Института прикладной математики им. М.В.Келдыша РАН, 1999, N75
10. Документы ISO/IEC JTC1/SC22/WG5/
11. Горелик А.М., Ушкова В.Л., Шура-Бура М.Р. Мобильность программ на Фортране. М.: Финансы и статистика, 1984
12. Горелик А.М., Ушкова В.Л. Средства поддержки надежности и мобильности программ в языках программирования - Программирование, 1986, N5, 42-52
13. Gorelik A.M., Ushkova V.L. Supporting Program Portability and Reliability in Programming Languages - Programming andComputing Software, July, 1987