Ханспетер Мессенбок
Никлаус Вирт

Различия между языками Oberon и Oberon-2

Hanspeter Moessenboeck, Niklaus Wirth (1993)
"Differences between Oberon and Oberon-2"
Institute for Computer Systems, ETH, Zurich, Technical Paper, July 1993

Перевод Р. Богатырев, 1995

Сведения об авторах

Ханспетер Мессенбок (Hanspeter Moessenboeck) — известный специалист в области компиляторов, языков и систем программирования, объектно-ориентированного программирования, программной инженерии; разработчик экспериментального языка Object Oberon, вместе с Никлаусом Виртом (Niklaus Wirth) является создателем языка Oberon-2. Он участвовал в ряде проектов, проводимых в Швейцарском Федеральном технологическом институте (Swiss Federal Institute of Technology), был профессором в департаменте компьютерных наук (Computer Science Department) в ETH (Цюрих, Швейцария). Он автор известной книги «Object-Oriented Programming in Oberon-2» (Springer-Verlag, 1993). В настоящее время работает профессором в университе Йоганна Кеплера в Линце (Johannes Kepler University, Linz, Austria).

Никлаус Вирт (Niklaus Wirth) — профессор Швейцарского Федерального технологического института (ETH) в Цюрихе, который Вирт закончил в 1958 г. и где получил специальность инженера в области электроники. Затем он продолжил свое обучение в Лавальском университете (Laval University) в Квебеке (Канада). В университете Калифорнии в Беркли (University of California at Berkeley, США) в 1963 г. Вирт защитил докторскую диссертацию. До 1967 г. работал доцентом на вновь образованном факультете компьютерных наук в Стэнфордском университете (Stanford University, США), где он разработал язык PL360 и, в сотрудничестве с рабочей группой IFIP Working Group 2.1, язык Algol-W. В том же 1967 г. становится доцентом в университете Цюриха (University of Zurich), а в 1968 г. переходит в ETH, где в период с 1968 по 1970 годы разрабатывает язык Pascal. Среди последующих проектов — разработка и реализация персонального компьютера Lilith, высокопроизводительной 16-разрядной рабочей станции с растровым дисплеем, создание языка Modula-2 (1978-1982 г.), и 32-разрядной рабочей станции Ceres (1984-1986 г.). Затем им были созданы языки Oberon и Oberon-2 (совместно с профессором Х.Мессенбоком), а также операционная система Oberon (1986-1989 г.). В 1984 г. профессор Вирт был удостоен почетной премии Алана Тьюринга (Turing Award), в 1989 г. — премии Max Petitpierre Prize, а также премии Science and Technology Prize от IBM Europe. Один из последних его проектов — система Lola System для разработки электронных схем (1995 г.).

Аннотация

В статье излагаются различия между языками Oberon и Oberon-2. Основное внимание уделено таким новым концепциям, вошедшим в язык Oberon-2, как связанные процедуры (type-bound procedures), симплексный экспорт (read-only export), открытые массивы в роли ссылочных базовых типов. В язык также возвращен оператор FOR. В конце работы представлен список изменений, внесенных в первоначальное описание языка Oberon-2.

Ключевые слова

Oberon, Oberon-2, связанные процедуры, симплексный экспорт.

Язык Oberon-2 является чистым расширением языка Oberon [Wir88]. В этом работе представлены сделанные расширения языка и в ней предпринята попытка осветить причины этих изменений. Таким образом, мы надеемся упростить читателю задачу классификации языка Oberon-2. За более детальной информацией о языке читателю нужно обратиться непосредственно к описанию языка.

Одной из важнейших целей языка Oberon-2 было стремление упростить объектно-ориентировванное программирование без нарушения концептуальной простоты языка Oberon. После трех лет использования Oberon'а и его экспериментального диалекта — языка Object Oberon [MeT89] — мы воплотили наш опыт в единую уточненную версию Oberon'а.

Новыми особенностями языка Oberon-2 служат связанные процедуры (type-bound procedures), экспорт переменных и полей записи только на чтение (read-only export), открытые массивы в роли ссылочных базовых типов, а также оператор WITH с вариантами. Помимо этого в язык был возвращен оператор FOR, который первоначально при переходе от Modula-2 к Oberon был изъят из языка.

Язык Oberon-2 является плодом многочисленных дискуссий, которые велись среди всех сотрудников Института компьютерных систем в ETH (Institute for Computer System at ETH). Особенно плодотворные идеи были высказаны Юргом Гуткнехтом (Juerg Gutknecht) и Йозефом Темплом (Josef Templ).

СВЯЗАННЫЕ ПРОЦЕДУРЫ

Процедуры могут быть связаны с типом запись или со ссылочным типом. Они эквивалентны методам в объектно-ориентированной терминологии. Присоединение процедуры выражается с помощью дополнительного параметра (играющего роль операнда, к которому применяется процедура, или же «получателя», как это называется в объектно-ориентированной терминологии).

```
TYPE
  Figure = POINTER TO FigureDesc;
  FigureDesc = RECORD
    x,y,w,h: INTEGER
  END;

PROCEDURE (f: Figure) Draw; BEGIN ... END Draw;

PROCEDURE (f: Figure) Move (dx,dy: INTEGER); BEGIN ... END Move;
```

Draw и Move связаны с типом Figure, из чего следует, что обе они могут быть применимы к объектам типа Figure. Эти процедуры рассматриваются как локальные по отношению к FigureDesc и к ним можно обращаться как к обычным полям записи, например:

```
f.Move(10,10),
```

если f — это переменная типа Figure.

Любая процедура, связанная с типом T, неявно также связана и со всеми расширениями типа T. Она может быть переопределена (перегружена) процедурой с тем же самым именем и точно таким же списком формальных параметров, как и та, что явным образом связана с расширением T. Взгляните на пример:

```
TYPE
  Circle = POINTER TO CircleDesc;
  CircleDesc = RECORD (FigureDesc)
    radius: INTEGER
  END;

PROCEDURE (c: Circle) Move (dx,dy: INTEGER); BEGIN ... END Move;
```

Тип Circle является расширением типа Figure. Процедура Move явным образом связана с Circle и переопределяет процедуру Move, которая была «унаследована» от Figure. Пусть «f» — это переменная типа Figure, а «c» — переменная типа Circle, тогда присваивание f := c меняет у переменной «f» ее динамический тип (ее тип) с Figure на Circle. В вызове

```
f.Move(10,10)
```

переменная «f» выполняет две функции: во-первых, она передается в качестве параметра-получателя процедуре Move, а во-вторых, ее динамический тип определяет то, какой вариант Move в действительности вызывается. Так как после присваивания f := c динамический тип переменной «f» уже Circle, то вызывается процедура Move, которая связана с типом Circle, а не та, что связана с типом Figure. Этот механизм называется динамическим связыванием (dynamic binding), поскольку динамический тип получателя используется для связывания имени процедуры с реальной процедурой.

Внутри переопределяющей (redefining) процедуры переопределяемая (redefined) процедура может вызываться с использованием значка ^, например:

```
f.Move^(dx,dy).
```

Аргументация. Мы отказались от введения концепции класса, а вместо этого заменили его на хорошо известную концепцию записи. Таким образом, класс — это просто тип запись со связанными с ним процедурами. Мы также отказались от дублирования в самой записи заголовков связанных процедур, как это сделано в других языках типа C++ и Object Pascal. Это позволяет сохранить записи короткими и избежать избыточности описаний. Ведь изменение в заголовке тогда повлекло бы за собой корректировку в двух разных местах программы, к тому же компилятору пришлось бы еще и проверять идентичность обоих заголовков. Если программист захочет увидеть конкретную запись со всеми связанными с ней процедурами, то он может для получения информации на экране или на листе бумаги воспользоваться специальным инструментом, который называется навигатор модулей (browser).

Процедуры, привязанные к типу, могут быть объявлены в произвольном порядке. Их можно даже перемежать процедурами, привязанными к другому типу. В языке Object Oberon, где все методы должны быть объявлены внутри соответствующего объявления класса, обнаружилось, что косвенная рекурсия между методами различных классов может поставить в затруднительное положение предварительные описания целых классов.

В языках типа Object Pascal и C++ переменные экземпляра (instance variable) для объекта-получателя «self» могут быть доступны как с квалификацией, так и без нее (например, можно написать и «x», и «self.x»). В этих языках подчас трудно определить, что перед вами — обычная переменная или же переменная экземпляра. Бывает еще сложнее, когда имя обозначает переменную экземпляра, которая унаследована от базового класса. По этой причине мы решили, что переменные экземпляра в языке Oberon-2 должны всегда квалифицироваться. Таким образом, удается избежать выбора между двумя семантически эквивалентными конструкциями, что на наш взгляд нежелательно для языков программирования.

В языке Oberon-2 получатель представляет собой явный параметр, а потому программист может выбирать для него характерное имя, которое обычно выглядит куда более выразительнее, чем предопределенное имя «self», что используется в других объектно-ориентированных языках. Явное объявление получателя проясняет ту картину, что объект, к которому применяется данная операция, передается этой операции в качестве параметра. Это обычно нельзя выразить в других объектно-ориентированных языках. К тому же такой подход четко следует духу Оберона — избегать любых скрытых механизмов.

В языке Object Oberon методы имеют в точности такой же синтаксис, как и обычные процедуры. В больших классах, где заголовка класса не видно рядом с заголовком метода, невозможно понять, является ли данная процедура обычной процедурой или же перед нами метод. В этом случае трудно также понять, к какому классу данный метод относится. В языке Oberon-2 тип параметра-получателя связанной процедуры обозначает тип, с которым процедура связана, поэтому никаких сомнений не возникает.

СИМПЛЕКСНЫЙ ЭКСПОРТ (READ-ONLY EXPORT)

В то время как в языке Oberon все экспортируемые переменные и поля записей могут быть изменены любым клиентным модулем, в языке Oberon-2 появилась возможность ограничивать использование экспортированной переменной или поля записи доступом только на чтение. Этот факт отмечается объявлением с использованием знака - , а не * . Знак «минус» говорит об ограниченном использовании данной переменной.

```
TYPE
  Rec* = RECORD
    f0* : INTEGER;
    f1- : INTEGER;
    f2  : INTEGER;
  END;

VAR
  a* : INTEGER;
  b- : Rec;
  c  : INTEGER;
```

Клиентные модули могут читать переменные «a» и «b», а также поля «f0» и «f1», поскольку все эти объекты экспортируются. Однако, изменять они могут лишь «a» и «f0», в то время как значения «b» и «f1» модифицировать не удастся. Только тот модуль, который экспортирует эти объекты, и может изменять их значения. (Даже если модуль-клиент объявляет приватную переменную типа Rec, ее поле «f1» остается с доступом только на чтение.) Поскольку переменная «b» доступна только на чтение, то и все ее компоненты также доступны только на чтение.

Причины введения симплексного экспорта заключаются в желании обеспечить более тонкую настройку в плане инкапсуляции информации. Инкапсуляция преследует две цели: во-первых, она помогает избавить клиентов от ненужных деталей, и во-вторых, гарантировать, что значения скрытых переменных изменяются лишь процедурами доступа внутри содержащего их модуля. Симплексный экспорт как раз и преследует вторую цель.

Открытые массивы

И в языке Modula-2, и в языке Oberon можно использовать открытые массивы в качестве параметров. Длина такого массива определяется длиной фактического параметра процедуры. В языке Oberon-2 открытые массивы могут не только быть объявлены в качестве типа формального параметра, но и как ссылочные базовые типы (pointer base type). В этом случае для размещения в динамической памяти открытого массива с произвольной длиной используется предопределенная процедура NEW.

```
VAR v: POINTER TO ARRAY OF INTEGER; ... NEW(v,100)
```

При этом массив v^ на этапе выполнения программы размещается в динамической памяти, причем длина его составляет 100 элементов: от v[0] до v[99].

Оператор WITH

В языке Oberon оператор WITH представляет собой локального привратника типа в виде:

```
WITH v: T DO S END;
```

Если динамический тип переменной «v» — это T, то выполняется последовательность операторов S, причем привратник типа v(T) применяется к каждому появлению переменной «v», другими словами, переменная «v» рассматривается так, как если бы она имела статический тип T. Если динамический тип переменной «v» не T, то программа аварийно завершается. В языке Oberon-2 оператор WITH может быть записан с разными вариантами:

```
WITH
  v: T0 DO S0 | v: T1 DO S1 ELSE S2 END;
```

Если динамический тип переменной «v» — это T0, то выполняется последовательность операторов S0, а сама переменная «v» рассматривается так, как если бы она имела статический тип T0. Если динамический тип переменной «v» — это T1, то выполняется последовательность операторов S1, а переменная «v» рассматривается так, как если бы она имела статический тип T1. В противном случае выполняется последовательность S2. Если ни один из вариантов не был выполнен и если ELSE-часть отстутствует, то программа аварийно завершается.

ОПЕРАТОР FOR

Хотя оператор FOR можно всегда выразить через оператор WHILE, все же иногда он более удобен, поскольку имеет лаконичную форму и завершение цикла в нем четко выражено. Его удобно использовать, когда требуется выполнить фиксированное число итераций подобно тому, как это часто нужно в случае работы с массивами. Оператор FOR записывается в виде:

```
FOR i := a TO b BY step DO statements END
```

Этот оператор эквивалентен следующей последовательности операторов:

```
temp := b; i := a; IF step > 0 THEH
  WHILE i <= temp DO statements; i := i + step END ELSE
  WHILE i >= temp DO statements; i := i + step END END;
```

ПРИЛОЖЕНИЕ

Список изменений, внесенных в язык Oberon-2

Здесь собраны изменения, вносимые в первоначальное описание языка Oberon-2.

Июль 1993 г.

6.4. Больше не требуется, чтобы переменные-указатели инициализировались значением NIL. Однако, реализации языка должны по запросу предоставлять соответствующую инициализацию указателей.

9.8. Изменена инициализация оператора FOR. Переменная «temp» инициализируется до переменной «v» и специфицируется тип «temp».

Это изменение было предложено Ником Уолшем (Nick Walsh) из Лондонского университета (City University, London) с тем, чтобы можно было корректно работать с оператором вида

```
FOR i := 0 TO i DO ... END.
```

10.3. Введена предопределенная процедура ASSERT.

Октябрь 1993 г.

6. Составной тип не может содержать самого себя.

11. Модуль не должен импортировать самого себя.

8.2.4. Сравниваемые друг с другом массивы литер должны в качестве ограничителя иметь литеру с кодом 0X.

10.3. Параметр «x» процедуры COPY должен быть либо строкой, либо в качестве ограничителя иметь литеру с кодом 0X.

A. Совместимость по выражению: сравниваемые массивы должны в качестве ограничителя иметь литеру с кодом 0X.

ЛИТЕРАТУРА

- [MeT89] Moessenboeck H., Templ J. (1989) «Object Oberon — A Modest Object-Oriented Language» // Structured Programming, Vol.10, No.4, p.199-207.
- [Wir88] Wirth N. (1988) «The Programming Language Oberon» // Software — Practice and Experience, Vol.18, No.7, p.671-690.