Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

НЕКОТОРЫЕ ЗАМЕЧАНИЯ ОТНОСИТЕЛЬНО ЯЗЫКА *ПАСКАЛЬ*

МАРГАРИТА Р. БАРНЕВА

Рассматриваются некоторые конструкции языка программирования *ПАСКАЛЬ*. Приводятся результаты анализа синтаксическо описания языка и возможная базисная система.

Первое описание языка *ПАСКАЛЬ* появилось в 1971 году [1]. В 1974 году было опубликовано пересмотренное сообщение об этом языке [2].

Язык *ПАСКАЛЬ* представляет развитие языка *АЛГОЛ-60* [3]. В отличии от *АЛГОЛ-68* [4], в котором языковые формы сильно усложнены, в языке *ПАСКАЛЬ* основные формы *АЛГОЛ-60* сохранились и включены дополнительные возможности для работы со структурными данными.

Простота языка *ПАСКАЛЬ* в известном смысле является реакцией на тенденцию сильного усложнения *АЛГОЛ-68*. Этим можно объяснить широкую популярность и множество реализации языка *ПАСКАЛЬ*.

1. Примечания относительно некоторых элементов языка. Возможность определять тип ⟨*subrange*⟩ в *ПАСКАЛЬ* позволяет осуществлять более полный контроль на языковом уровне. Так, например, переменная K в *ПАСКАЛЬ* можно определить описанием

K: 1..50

и тогда переменная K будет принимать целые значения с 1 по 50. В языке *АЛГОЛ-60* (или соответственно в *ФОРТРАН*) K можно определить только как переменной целого типа, и программист сам должен следить, чтобы значения переменной не выходили из интервала [1, 50]

Действительно, в *АЛГОЛ-60* в случае

array *a* [1 : 50]

следует, что индексное выражение K должно принимать значения в интервале [1,50], но так как K может участвовать в других частях программы, это ограничение не относится к переменной K.

Удобством языка *ПАСКАЛЬ* является обстоятельство, что программы оформляются в виде подпрограмм и могут использовать формальные параметры, относящиеся к входно-выходным файлом. Целесообразно расширить список этих параметров, охватывая другую информацию, связанную с выполнением программы, например, максимальное время выполнения программы, требуемая память и т. д.

В программах на языке *ПАСКАЛЬ* можно пользоваться блоковой структурой только искусственным путем — при помощи аппарата процедур.

Несмотря на то, что синтаксическое определение конструкции ⟨*expression*⟩ позволяет использовать как аргумент переменную любого типа, на самом деле над переменными структурного типа можно производить слишком ограниченное число операций.

СЕРДИКА Българско математическо списание. Том 5, 1979, с. 34—38.

Структурный тип ⟨*class type*⟩ для описания данных, связанных в конечном графе, в ревизированном сообщении не включен. Тип ⟨*pointer*⟩ дает те же возможности, но он менее естествен.

Возможность применять стандартные функции *pred* и *succ* к аргументам типа *real* не имеет смысла, так как результат зависит от реализации языка.

Запрещение определения скалярных типов с общими элементами не гармонирует с возможностью определять поддиапазоны чисел целого типа с непустым сечением.

Пересмотренное сообщение [ ] включает дополнения, относящиеся к операциям ввода-вывода, но они не нашли отражения в синтаксисе языка.

Принятая в синтаксисе языка *ПАСКАЛЬ* последовательность при написании разных частей программы, а именно:

⟨*label declaration part*⟩
⟨*constant definition part*⟩
⟨*type definition part*⟩
⟨*variable declaration part*⟩
⟨*procedure and function declaration part*⟩
⟨*statement part*⟩

так, что каждая часть может пользоваться только своими объектами и объектами предыдущих частей, облегчает реализацию трансляторов. Недопустимость „цитирования вперед", однако, не относится к объектам внутри отдельных программных частях.

Включение возможности предварительного описания определенных типов целесообразно, когда в программе будут использоваться несколько переменных одного типа и ввиду того, что облегчается трансляция. Однако эта концепция реализована неполностью, так как допустимо описание типа в объявлении переменной, например:

var *a*: **array** [1..50] **of** *integer*.

При определении тип ⟨*pointer*⟩ в ⟨*type definition part*⟩ невозможно обойти „цитирования вперед", так как тип ⟨*pointer*⟩ всегда связан с определенным структурным типом. Например:

type *link* = *person*; ...
person = **record** ... *next* : *link*; ... **end**;

Определение типов можно описать еще следующим способом:

type *person* = **record** ... *next* : *person*; ... **end**;
link = *person*;...

который приводит к своеобразной рекурсии. Определение тип ⟨*pointer*⟩ в первоначальной версии языка создавало неудобство аналогичного характера.

Интересные замечания относительно некоторых двусмысленностей и неточностей в языке *ПАСКАЛЬ* опубликованы в [5].

2. Примечания относительно синтаксического описания языка. Синтаксис языка *ПАСКАЛЬ* описывается при помощи металингвистических правил (формул). Применяется простая модификация метаязыка *Бакуса—Наура*. Начальный (помеченный) символ грамматики — *программа* — базируется на множество терминальных символов и металингвистичных переменных (нетерминальных символов).

Синтаксическое описание первоначальной версии языка *ПАСКАЛЬ* содержит 97 металингвистических формул. В пересмотренном сообщении отброшены 9 из них и добавлены 22 новые, так, что общее число правил — 110. Введены дополнительно следующие металингвистичные переменные:

⟨*digit sequence*⟩, ⟨*unsigned integer*⟩, ⟨*unsigned real*⟩, ⟨*unsigned number*⟩, ⟨*string*⟩, ⟨*structured type*⟩, ⟨*case label list*⟩, ⟨*unpacked structured type*⟩, ⟨*base type*⟩, ⟨*element list*⟩, ⟨*file buffer*⟩, ⟨*unlabelled statement*⟩, ⟨*empty statement*⟩, ⟨*record variable list*⟩, ⟨*block*⟩, ⟨*label declaration part*⟩, ⟨*program*⟩, ⟨*program heading*⟩ ⟨*empty*⟩, ⟨*character*⟩.

Перечисленные ниже результаты относятся к синтаксису пересмотренного сообщения языка.

В синтаксисе используются 120 терминальных символов и 113 металингвистических переменных, причем 2 из них — ⟨*empty*⟩ и ⟨*character*⟩ не определены. Металингвистическая переменная ⟨*case label list*⟩ определяется два раза идентичным образом. Определение некоторых металингвистических переменных можно написать более коротко. Например, ⟨*adding operator*⟩ определен так:

⟨*adding operator*⟩ :: = + | − | **or**,

а можно определить двумя альтернативами, используя, что ⟨*sign*⟩ :: = + | −.

Так как в языке введена металингвистическая переменная ⟨*constant identifier*⟩, то определение ⟨*constant definition*⟩ :: = ⟨*identifier*⟩ = ⟨*constant*⟩ лучше заменить формулой:

⟨*constant definition*⟩ :: = ⟨*constant identifier*⟩ = ⟨*constant*⟩.

В формуле, определяющей ⟨*actual parameter*⟩, альтернатива ⟨*variable*⟩ не нужна, так как ⟨*variable*⟩ есть выражение.

Точка в конце металингвистической формулы, определяющей ⟨*program*⟩, не нужна.

В пересмотренном сообщении включена дополнительно металингвистическая переменная ⟨*simple type*⟩ для короткого определения индексного и базисного типа. В использованном определении ⟨*simple type*⟩ :: = ⟨*scalar type*⟩ | ⟨*subrange type*⟩ | ⟨*type identifier*⟩, очевидно, исходя из семантики и синтаксиса языка, имеется в виду, что ⟨*type identifier*⟩ обозначает или скалярный тип, или тип поддиапазона. Но эта металинвистическая формула некорректна, так как ⟨*type identifier*⟩ может обозначать произвольный структурный тип.

3. Выбор базисной системы. Известно, что при разработке синтаксически управляемых трансляторов полезным может оказаться выбор базисной системы символов (базис) [6]. Задача о выборе базиса состоит в следующем.

Пусть задан язык программирования, грамматика которого представляет упорядоченную четверку (T, V, F, P), где

T — множество терминальных символов,

V — множество нетерминальных символов,

F — множество металингвистических правил,

P — помеченный символ.

Множество B $\{b_1, b_2, \ldots b_n\}$, $B \subset T \cup V$, является базисом, если существует G $\{g_1, g_2, \ldots g_n\}$, $G \subset F$ так, что помеченный символ P можно определить элементами B и правилами из G.

Базис позволяет разложить синтаксический анализ на два уровня: анализ на основе терминальных символов до уровня базисных и последующий анализ, с уровня базисных символов до помеченного символа.

Выбор подходящего базиса зависит от ряда соображений.

В [7] рассмотрены алгоритмы определения базиса и приведены результаты исследования языка *АЛГОЛ-60*.

Выбор базисной системы для языка *ПАСКАЛЬ* был сделан при помощи программы на языке *ФОРТРАН* для ЭВМ Минск-32. Программа вводит множество символов C $\{c_1, c_2, \ldots c_n\}$, $C \subset V$ и проверяет, есть ли множество T', $T' \subset T$ такое, что множество $B = C \cup T'$ образует базисную систему.

Проведенные эксперименты показали, что базис образуют нетерминальные символы:

⟨*identifier*⟩
⟨*digit sequence*⟩
⟨*constant*⟩
⟨*variable*⟩
⟨*expression*⟩
⟨*assignment statement*⟩
⟨*empty*⟩

вместе со следующими 38 терминальными символами:

( ),; = .. [ ] : ↑ **array begin case const do downto else end file for function goto if label of packed procedure record repeat set then type to until var while with.**

При такой системе базисных символов множество G содержит 68 металингвистических правил.

Интересно отметить, что в *АЛГОЛ-60* символ ⟨*expression*⟩ было невозможно включить в систему базисных символов. Это объясняется большей независимостью отдельных конструкций языка *ПАСКАЛЬ* и более простой структурой понятия ⟨*expression*⟩.

Помимо основной цели, программа определяет группы металингвистических переменных, использованных в качестве синонимов. В языке *ПАСКАЛЬ* существуют девять таких групп:

1) ⟨*unsigned integer*⟩, ⟨*digit sequence*⟩, ⟨*label*⟩
2) ⟨*constant identifier*⟩, ⟨*identifier*⟩, ⟨*type identifier*⟩, ⟨*variable identifier*⟩, ⟨*field identifier*⟩, ⟨*procedure identifier*⟩ ⟨*control variable*⟩, ⟨*function identifier*⟩, ⟨*result type*⟩, ⟨*entire variable*⟩
3) ⟨*index type*⟩, ⟨*simple type*⟩, ⟨*base type*⟩
4) ⟨*component type*⟩, ⟨*type*⟩
5) ⟨*case label*⟩, ⟨*constant*⟩
6) ⟨*record variable*⟩, ⟨*variable*⟩, ⟨*file variable*⟩, ⟨*pointer variable*⟩, ⟨*array variable*⟩
7) ⟨*empty statement*⟩, ⟨*empty*⟩
8) ⟨*initial value*⟩, ⟨*expression*⟩, ⟨*final value*⟩
9) ⟨*statement part*⟩, ⟨*compound statement*⟩.

Как известно, синонимные символы имеют отношение только к семантике языка и облегчают его понимание. Удаление синонимов сократит существенно синтаксическое описание языка.

Использованную программу можно применять к произвольному языку, синтаксис которого описан при помощи метаязыка Бэкуса.

ЛИТЕРАТУРА

1. N. Wirth. The Programming Language Pascal. *Acta Informatica*, **1**, 1971, 35—63. (Н. Вирт. Язык программирования ПАСКАЛЬ. *Алгоритмы и организация решения экономических задач*, Москва, 1974).

2. K. Jensen, N. Wirth. *PASCAL*. User manuel and report. (*Lecture notes in computer Science*, 18,) Berlin, 1974.
3. P. Naur et al. Report on the algorithmic language *ALGOL 60*. *Numer math.*, **2**, 1960, 106—136. (Сообщение об алгоритмическом языке Алкал-60. *Ж. Вычисл. мат. и мат. физ.* **1**, 1961, 308—342.)
4. A. van Wijngarden (Editor). B. J. Mailoux, J. E. L. Peck, C. H. A. Koster. Report on the algorithmic language ALGOL-68. Amsterdam, 1969. (А. ван. Аайнгарден, Б. Ж. Майу, Дж. Е. Л. Пек К. Х. А. Костер. Алгоритмичният език АЛГОЛ-68. София, 1971).
5. J. Welsh, W. Sneeringer, G. Hoare. Ambiguities and unsecurities in Pascal. *Software — practice and experience*, **7**, 1977, 685—696.
6. L. Bollet. L.écreture des compilateurs. *Rev. franc. traitement inform.*, **9**, 1966, 47—73, 129—158.
7. П. Бърнев, В. Томов, М. Бърнева. Преобразуване на синтаксиса на алгоритмични езици във връзка с построяване на синтаксически управляеми транслатори. *Известия Мат. инст. БАН*, **13**, 1972 105—129.

Единый центр математики и механики
1090 София П. Я. 373

Поступила 22. 5. 1978