

Приложение В: Отличия Компонентного Паскаля от Паскаля

© Английский оригинал: Oberon microsystems, 1994-2001.

© Перевод на русский язык: Ф.В.Ткачев, апрель 2001, март 2009.

Исключенные средства

- **Типы-диапазоны**

Используйте один из стандартных целых типов.

- **Перечислимые типы**

Используйте вместо них целые константы.

- **Произвольные диапазоны для массивов**

Массивы теперь всегда определены над целым диапазоном 0..max-1.

Пример

A = ARRAY 16 OF INTEGER (* разрешены индексы из диапазона 0..15 *)

- **Нет общих множеств**

Тип SET теперь описывает набор целых чисел, который может включать элементы 0..31.

- **Нет явного оператора DISPOSE**

Неиспользуемая более память автоматически собирается сборщиком мусора. Вместо DISPOSE, просто присвойте переменной значение NIL.

- **Нет вариантных записей**

Используйте расширение (расширенное переопределение) записей.

- **Нет упакованных структур**

Используйте типы SHORTCHAR или BYTE для значений, уместящихся в байт.

- **Нет GOTO**

- **Нет стандартных функций PRED и SUCC**

Используйте DEC и INC для целых значений.

- **Нет встроенных средств ввода/вывода**

Нет файловых типов. Ввод/вывод обеспечивается библиотечными процедурами.

Измененные средства

- **Стандартная процедура ENTIER вместо ROUND**

- **Синтаксис для констант типа REAL**

3.0E+4, но не 3.0e+4

- **Синтаксис для объявлений указательных типов**

P = POINTER TO R

вместо

P = ^R

- **Синтаксис для оператора CASE**

"|" вместо ";;" в качестве разделителя случаев.
Предложение ELSE.

Пример

```
CASE i * 3 - 1 OF
  0: StdLog.String("нуль")
| 1..9: StdLog.String("от единицы до девяти")
| 10, 20: StdLog.String("десять или двадцать")
ELSE StdLog.String("что-то еще")
END
```

- **Имя процедуры должно быть повторено**

Пример

```
PROCEDURE DrawDot (x, y: INTEGER);
BEGIN
END DrawDot;
```

- **Большие и малые буквы различаются**

Пример "proc" не то же самое, что "Proc".

- **Синтаксис литерных цепочек**

Литерные цепочки-константы заключаются между " или между '. В одной цепочке не может быть одновременно одиночных и двойных кавычек. Литерные цепочки-константы единичной длины могут присваиваться литерным переменным.

Пример

```
"That's great"  'Write "hello world" to the screen'
ch := "x"
ch := 'x'
```

- **Комментарии**

Комментарии заключаются между (* и *) и могут быть вложены.

- **Скобки для множеств**

Константы-множества задаются между { и } вместо [и].

Пример {0..2, 4, j..2 * k}

- **Синтаксис функций**

Используйте ключевое слово PROCEDURE также и для функций вместо FUNCTION.

Процедуры, возвращающие значение, всегда должны иметь (возможно пустой) список параметров в своих объявлениях и в вызовах.

Результат функции возвращается явно оператором RETURN, вместо присваивания имени функции.

Пример

```
PROCEDURE Fun (): INTEGER;
BEGIN
  RETURN 5
END Fun;
```

вместо

```
FUNCTION Fun: INTEGER;  
BEGIN  
    Fun := 5  
END;
```

n := Fun() вместо n := Fun

• Объявления

Последовательность объявлений теперь имеет вид
{ ConstDecl | TypeDecl | VarDecl } { ProcDecl | ForwardDecl }
вместо
[ConstDecl] [TypeDecl] [VarDecl] {ProcDecl}.

Упреждающее (forward) объявление необходимо, если процедура используется до ее определения.

Пример

```
PROCEDURE ^ Proc;  
вместо  
PROCEDURE Proc; FORWARD;
```

• Процедурные типы

Процедуры могут быть не только переданы в качестве параметров, но и присваиваться переменным процедурных типов.

Пример

```
TYPE P = PROCEDURE (x, y: INTEGER);  
VAR v: P;  
v := DrawDot;    (* присваивание *)  
v(3, 5);    (* вызов DrawDot(3, 5) *)
```

• Явные END вместо составных операторов

BEGIN может появляться только перед последовательностью операторов, но не внутри ее.
IF, WHILE и LOOP всегда заканчиваются ключевым словом END.

• Оператор WITH

Оператор WITH является охраной типа, действующей в соответствующем программном фрагменте, он не подразумевает наличие скрытой переменной и не открывает новый диапазон видимости для переменных. См. детали в описании языка.

• ELSIF

Операторы IF могут иметь несколько ветвей.

Пример

```
IF name = "top" THEN  
    StdLog.Int(0)  
ELSIF name = "bottom" THEN  
    StdLog.Int(1)  
ELSIF name = "charm" THEN  
    StdLog.Int(2)  
ELSIF name = "beauty" THEN  
    StdLog.Int(3)  
ELSE  
    StdLog.String("strange")  
END
```

• BY вместо только DOWNT0 в FOR

Циклы FOR могут использовать любое константное значение в качестве приращения (положительного или отрицательного).

Пример

```
FOR i := 15 TO 0 BY -1 DO StdLog.Int(i, 0) END
```

- **Логические выражения используют «сокращенное» вычисление**

Вычисление логического выражения прекращается, как только его результат определен.

Пример

Следующее выражение не вызывает ошибки при выполнении, когда p = NIL:

```
IF (p # NIL) & (p.name = "quark") THEN
```

- **Константные выражения**

В объявлениях констант допустимы не только буквальные константы, но и константные выражения.

Пример

```
CONST  
    zero = ORD("0");  
    one = zero + 1;
```

- **Разные операции**

используется вместо <> для проверки на неравенство.

& используется вместо AND для логической конъюнкции.

~ используется вместо NOT для логического отрицания.

- **Явное преобразование к меньшему типу с помощью SHORT**

Включение типов для числовых типов позволяет присваивать значения меньшего типа переменной большего типа. Присваивание в обратном направлении должно использовать стандартную процедуру SHORT.

Пример

```
int := shortint;  
shortint := SHORT(int)
```

Новые средства

- **Шестнадцатеричные числа и литеры**

Пример

```
100H    (* десятичное 256 *)  
0DX     (* возврат каретки *)
```

- **Дополнительные числовые типы**

Добавлены типы LONGINT, SHORTINT, BYTE, SHORTREAL.

- **Симметрическая разность множеств**

Множества могут вычитаться.

- **Новые стандартные процедуры**

Добавлены новые стандартные процедуры INC, DEC, INCL, EXCL, SIZE, ASH, HALT, ASSERT, LEN, LSH, MAX, MIN, BITS, CAP, ENTIER, LONG и SHORT.

- **LOOP с EXIT**

Имеется новый оператор цикла с явным оператором выхода. См. детали в сообщении о языке.

- **ARRAY OF CHAR могут сравниваться**

Литерные массивы могут сравниваться с помощью операций =, #, <, >, <= и >=.

- **Открытые массивы, многомерные массивы**

Можно определять массивы, не указывая их размера, возможно, с несколькими измерениями.

Пример

```
VAR a: POINTER TO ARRAY OF CHAR;  
NEW(a, 16)
```

```
PROCEDURE ScalarProduct (a, b: ARRAY OF REAL; VAR c: ARRAY OF REAL);
```

```
TYPE Matrix = ARRAY OF ARRAY OF REAL;  
PROCEDURE VectorProduct (a, b: ARRAY OF REAL; VAR c: Matrix);
```

- **Разыменование указателей не обязательно**

Операция разыменования ^ может быть опущена.

Пример

```
root.next.value := 5  
вместо  
root^.next^.value := 5
```

- **Модули**

Модули суть единицы компиляции, упрятывания информации, а также загрузки. Упрятывание информации -- одна из главных черт объектно-ориентированного программирования. Возможны разные уровни упрятывания информации: полное упрятывание, экспорт только для чтения/реализации, полный экспорт. См. детали в сообщении о языке.

- **Расширенное переопределение (расширение) типов**

Типы записей (указательные типы) могут переопределяться, обеспечивая таким образом полиморфизм. Полиморфизм -- одно из главных средств объектно-ориентированного программирования.

- **Методы**

Процедуры могут быть связаны с типами записей (указательными типами), таким образом обеспечивая позднее связывание [late binding]. Позднее связывание является одним из главных средств объектно-ориентированного программирования. Такие процедуры еще называются *методами*.

- **Операция с цепочками литер**

Литерная цепочка, содержащаяся в литерном массиве, может быть выбрана посредством селектора \$.

- **Атрибуты записей**

По умолчанию записи не могут быть расширены (переопределены), но могут быть помечены как EXTENSIBLE, ABSTRACT или LIMITED.

- **Атрибуты методов**

По умолчанию методы не могут быть расширены (переопределены), но могут быть помечены как EXTENSIBLE, ABSTRACT или EMPTY. Вновь вводимые методы должны быть помечены как NEW.