Введение в Pascal

Язык схем алгоритмов (ЯСА) не имеет реализации и поэтому переходим на алгоритмический язык (АЯ) Pascal. Сначала мы рассмотрим некоторое подмножество АЯ Pascal, которое по своим выразительным возможностям примерно соответствует ЯСА.

1. Переменные и константы

Имена переменных должны состоять только из лат. букв и цифр и начинаться с буквы.

Вводится понятие типа переменной. ***Тип переменной определяет множество значений, которые может принимать переменная, и множество операций, применимых к ним***. Пока мы рассмотрим только два типа: Integer - ***целые*** и Real - ***вещественные.***

Целые и вещественные переменные имеют разное внутреннее представление в ЦВМ, и это только подчёркивает их приципиальное различие. Надо исходить из того, что вещесвенные переменные, в отличие от целых, представляют значения приблизительно, хотя м.б. и с высокой точностью. О вычислениях с вещественными значениями можно говорить также как о вычислениях с заданным количеством значащих цифр. Пример вычислений с двумя значащими цифрами.

(6.2 - 6.1) * 5.0 = 0.10 * 5.0 =0.50

6.2 * 5.0 - 6.1 * 5.0 = 31 - 30 = 1.0

Получилось так, что (a - b)*c <> a*c - b*c ?! Этот пример показывает, что обычные правила арифметики на вещественные числа не распространяются. Поэтому надо быть осторожным, например, при проверке двух вещественных значений на равенство - в большинстве случаев следует поступать так: Abs(x1-x2) <= eps, где eps определяет заданную точность сравнения.

Тип Integer

Область значений: целые числа.
Операции: +, -, *, **div**, **mod**. a **mod** b = a - (a **div** b)*b
Примеры записи констант: 137, -1024.

Тип Real

Область значений: вещ. числа.
Операции: +, -, *, /.
Примеры записи констант: 0.1, 2.34, -5e-3, 2.5e2.

2. Арифметические выражения

Действуют схемы: i + i -> i, r + i -> r, r + r -> r. Аналогично и для операций - и *. Операции **div** и **mod** применимы только к целочисленным операндам и дают целочисленный результат. Операция / всегда даёт вещественный результат.

3. Метаязык для описания языка программирования

Для определения синтаксиса АЯ Pascal Вирт применил ***синтаксические диаграммы Вирта***, как теперь их уже принято называть. Мы же используем равномощный способ - ***расширенную форму Бэкуса-Наура***. Этот способ не столь нагляден, но зато более компактен. Это метаязык, который можно применить для описания различных ЯВУ. Определим элементы этого языка.

Нетерминальный символ - обозначает некоторое общее понятие определяемого языка. Наименование понятия заключают в угловые скобки. Примеры: <Оператор>, <Целое число>, <Имя>.
Терминальный символ - обозначает конкретный элемент определяемого языка. Его записывают в непосредственном виде. Примеры: 137, **begin**, +.
Начальный символ - нетерминальный символ, обозначающий самое общее понятие. Для ЯВУ обычно это <Программа>.
Правило - показывает, как определяется нетерминальный символ через другие символы языка. Пример:
<Оператор присваивания> ::= <Имя пер.> := <Ар. выр.>.
Знак "::=" читается как "это есть". Он делит правило на две части - левую (определяемый нетерминальный символ) и правую (сентенциальную форму - последовательность любых символов). Система таких правил и определяет язык. Кроме того, в правой части можно использовать некоторые знаки, которые служат для сокращения записи.
| - символ альтернативы. Позволяет несколько правил с одинаковой левой частью записать как одно правило с альтернативами. Пример: <Булевская константа> ::= True | False .
[] - необязательная часть формы. Пример: <Элемент> ::= <Выражение> *[* ..<Выражение> *]* .
{} - итерация. Повторяющаяся часть формы, в том числе и нуль раз. Пример: <Список> ::= <Элемент> *{* , <Элемент> *}* .

4. Оператор присваивания

<Оператор присваивания> ::= <Имя пер.> := <Ар. выр.>

Совместимость типов по присваиванию: тип переменной должен строго совпадать с типом значения выражения. Единственное исключение: r := i (***неявное преобразование типа***). Для явного преобразования r -> i предусмотрены стандартные функции Round(r) -> i и Trunk(r) -> i. Пример. Trunk(-1.9) -> -1.

5. Логические выражения

<Лог. выр.> ::= <Ар. выр.> <Оп. ср.> <Ар. выр.>

<Оп. ср.> ::= = | <> | < | <= | > | >=

Логическое выражение может иметь одно из двух значений - False (ложь) и True (истина).

6. Управляющие конструкции

Центральное понятие - ***оператор***. Различают ***основные*** и ***структурные*** операторы.
Последние могут включать в свой состав другие операторы.
Основные операторы: ***присваивания***, ***пустой***, ***вызова процедуры***.
Структурные операторы: ***составной оператор***, ***оператор*** **if**, ***оператор*** **while**, ***оператор*** **repeat**.

<Составной оператор> ::= **begin** <Оператор> *{* ; <Оператор> *}* **end**

<Оператор **if**> ::= **if** <Лог. выр.> **then** <Оператор> *[* **else** <Оператор> *]*

<Оператор **while**> ::= **while** <Лог. выр.> **do** <Оператор>

<Оператор **repeat**> ::= **repeat** <Оператор> *{* ; <Оператор> *}* **until** <Лог. выр.>

Оператор вызова процедуры пока мы будем использовать только для вызова стандартных процедур ввода-вывода.

7. Стандартные процедуры ввода-вывода

Стандартными или ***предопределёнными*** принято называть те элементы языка, которые определены вместе с языком. Естественно, что язык также предоставляет необходимые средства для определения нестандартных элементов соответствующего вида.

Стандартные операции ввода-вывода реализуют передачу данных между ***внешним устройством*** (ВУ) и ***основной памятью*** (ОП). Пока мы рассмотрим только ***текстовый ввод-вывод***.

Read(<Список ввода>)
ReadLn *[*(<Список ввода>) *]*
Write(<Список вывода>)
WriteLn *[*(<Список вывода>) *]*

Список ввода должен состоять из имён переменных. Стандартное устройство ввода - клавиатура. Данные в текстовом файле должны разделяться пробелами или концами строк.

Список вывода может состоять из арифметических выражений и строковых констант. Стандартное устройство вывода - монитор.

ReadLn и WriteLn отличаются от Read и Write тем, что выполняют перевод строки после передачи данных.

8. Структура программы

<Программа> ::= <Заголовок> <Блок> .

<Заголовок> ::= **program** <Имя программы>;

<Блок> ::= *{* <Раздел описаний> *}* <Раздел операторов>

<Раздел описаний> ::=
var <Имя пер.> *{*, <Имя пер.> *}* : <Имя типа> ;
{ <Имя пер.> *{*, <Имя пер.> *}* : <Имя типа> ; *}*

<Раздел операторов> ::= <Составной оператор>

9. Стандартные функции

Abs(i) -> i, Abs(r) -> r, Sqr(i) -> i, Sqr(r) -> r, Sqrt(r), Sin(r), Cos(r), Arctan(r), Ln(r), Exp(r). Углы измеряются в радианной мере.

10. Булевский тип

Имя типа: Boolean.
Значения: False и True.
Операции: **not**, **and**, **or**, <, <=, >, >=, =, <>.

Булевский тип позволяет обобщить понятие логического выражения на основе исчисления высказываний. Булевские операции **not**, **and** и **or** соответствуют операциям ~, & и | исчисления высказываний.

При отсутствии скобок в выражении порядок операций определяется их приоритетом. Приоритеты операций АЯ Pascal в порядке убывания:
1) **not**
2) * / **div mod and**
3) + - **or**
4) < <= > >= = <>

Переменные и константы типа Boolean нельзя использовать в списках ввода-вывода. При выводе можно использовать функцию Ord(x), которая возвращает порядковый номер значения переменной x (Ord(False) -> 0, Ord(True) -> 1). Для ввода можно использовать арифметические значения 0 и 1 с последующим выполнением оператора b := i = 1, что эквивалентно оператору **if** i = 1 **then** b := True **else** b := False .

Пример. Переменной p присвоить значение 1, если зачение x принадлежит отрезку (a,b). Решения:
1) **if** (x >= a) **and** (x <= b) **then** p := 1 **else** p :=0;
2) p := ord((x >= a) and (x <= b)).

Стадартная булевская функция Odd(i) -> b ("i - нечётное число").

Короткая и ***полная схема*** вычисления логических выражений.

11. Литерный тип

Имя типа: Char.
Значения: литеры кодовой таблицы ЦВМ.
Примеры констант: 'a', 'A', 'n', 'я', '&', '+', '0', '5' и т.д.
Операции: могут входить в список ввода-вывода стандартных процедур.
Стандартные функции: Ord(c) -> i, Chr(i) -> c.

Обязательные свойства кодовой таблицы ЦВМ:
1) Буквы латинского алфавита упорядочены - Ord('A') < Ord('B');
2) Цифры упорядочены и расположены плотно - Ord('5') = Ord('4') + 1 .

Преобразование литеры Ch в соответствующее числовое значение D и наоборот:
D := Ord(Ch) - Ord('0'); Ch := Chr(D + Ord('0')) .

Составим программу CharCode, которая позволяет определить код литеры, поступившей на её вход:

```
program CharCode;
var Ch: Char;
begin
Write('Char: '); ReadLn(Ch);
WriteLn('Ord(''',Ch,''') = ', Ord(Ch));
end.
```

Пример диалога программы CharCode:

```
Char: A
Ord('A') = 65
```

12. Классификация основных типов

Выше мы познакомились с типами Integer, Real, Boolean и Char. Все они являются стандартными. Кроме того их также принято называть ***основными типами***, подчёркивая тем самым, что они могут служить элементами для построения сложных типов, которые принято называть ***структурными***.

Все основные типы относятся к ***скалярным типам***. Переменная скалярного типа представляет только одну величину.

Существует также понятие ***ординального типа***. К ординальным относятся типы, к которым применима функция Ord. Это Integer, Boolean и Char. К ним применимы также функции Pred(x) и Succ(x), возвращающие соответственно предыдущее и последующее значение. Пример: Succ(4) = 5.

Внимание! Тип Real не является ординальным. Для значений этого типа в принципе не может быть понятия порядкового номера значения, а также предыдущего или следующего значения.

В заключение этого раздела составим программу BoolOp, которая выводит таблицу истинности для основных логических операций.

```
program BoolOp;
var  i  : Integer;
     a,b: Boolean;
begin
```

```
WriteLn('a  b  not a  a and b  a or b');
Writeln('---------------------------');
i:=0;
repeat
    a:=i mod 2 = 1; b:=i div 2 = 1;
    WriteLn(Ord(a), Ord(b):3,
            Ord(not a):5, Ord(a and b):8, Ord(a or b):9);
    i:=i+1;
until i>3;
end.
```

Выход программы BoolOp:

```
a  b  not a  a and b  a or b
---------------------------
0  0    1       0        0
0  1    1       0        1
1  0    0       0        1
1  1    0       1        1
```
