Система программирования PascalABC.NET: новые возможности

Михалкович С.С.
Южный федеральный университет,
факультет математики,
механики и компьютерных
наук
miks@math.sfedu.ru

Доклад на учебно-методической конференции
«Использование системы программирования PascalABC.NET и

К оглавлению

Русские идентификаторы

В PascalABC.NET можно использовать русские идентификаторы. После небольших переопределений программа преображается.

Использовать – дело вкуса

Русские идентификаторы

```
procedure Вывод(число: вещ) := Println(число);

type
  цел = integer;
  вещ = real;

begin
  var количество: цел := 10;

  var сумма: вещ := 0.0;
  for var i:=1 to количество do
    сумма += 1/i;

  Вывод(сумма);
end.
```

Описание переменных

- Переменные обычно описываются в блоке **begin** – **end**
- При описании переменную можно инициализировать выражением
- Если при описании переменную инициализировать выражением, то тип можно не писать – он **автоматически выводится** по типу выражения

Ко

Д

Нагромождение глобальных переменных – признак стиля старого Паскаля

```
var x: real; // В PascalABC.NET глобальные переменные описываются редко

begin
  var a1: integer;
  var a2 := 555;              // a2 получает тип integer по типу значения 555
  var r := 3.14;              // r получает тип real по типу значения 3.14
  var s := 'PascalABC.NET'; // s получает тип string

  var r1 := r + a2;           // r1 получает тот же тип, что и выражение r + a2
  var m := Min(2,3);          // m получает тип, возвращаемый функцией Min
  var d := DateTime.Now;      // d получает тип, возвращаемый DateTime.Now
end.
```

В PascalABC.NET переменные описываются по мере необходимости

Тип переменной автовыводится. Его надо указывать только в редких случаях для лучшей читаемости. Но в отличие от Python **тип** здесь – **статический**, не меняется по ходу программы

Стандартные функции

- Стандартные функции **Min**, **Max** и стандартную процедуру **Swap**
  удобно использовать в различных алгоритмах

Код

```
begin
  var (a,b,c) := (4,5,3);
  var min3 :=
  Min(Min(a,b),c);

  var a := // Пузырьковая сортировка
  ArrRandomInteger(10);
  for i var := 0 to a.High-1 do
  for i var := a.High downto i+1
  for  j if do
      Swap(a[j],a[j-1] then
      a[j-1]);
end.
```

Методы стандартных типов

- Стандартные типы integer, real и т.д. содержат большое количество стандартных методов и констант

Код

```
begin
  var i: integer := 56;
  var s: string := i.ToString;  // преобразование в строку

  if i.IsEven and i.InRange(10,99) then
    Println('Нечётное двузначное');

  i := integer.MaxValue;  // максимальная константа целого типа
end.
```

Цикл loop

Цикл loop используется в случаях, когда номер повторения цикла не важен

Пример 1. Геометрическая прогрессия

```
begin
  var n :=
  11; var a
  := 1; var q
  := 2;n do
  loop
  begin
    Print(a)
    a := a *
    q
  end;
```

```
1 2 4 8    32 64 128 256 512
16              1024
```

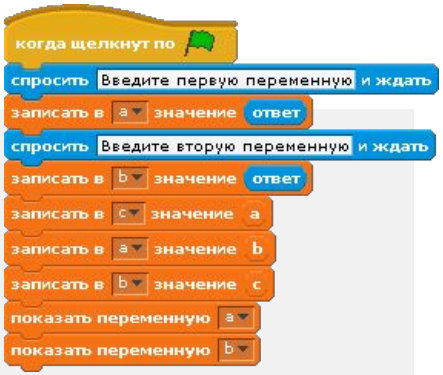
Пример 2. Сумма квадратов нечетных двузначных

```
begin
  var s := 0;
  var x :=
  11; loop 45
  do begin
    s := s +
    x; x := x
    + 2
  end;
  s.Print
```

```
2475
```

Множественное присваивание

- Можно присваивать сразу нескольким переменным
- Переменные из левой части получают значения из правой части **одновременно**
- Можно инициализировать сразу несколько переменных при описании. Их типы выводятся автоматически по типам инициализирующих выражений



Код

```
begin
  var (a,b,c) := (3,5,4);
  var (имя, возраст) := ('Петя', 10);

  (a,b) := (b,a); // перемена значений местами

  (a,b,c) := (b,c,a); // циклический сдвиг значений: 5 4 3
end.
```

К оглавлению

Примеры использования множественного присваивания

Множественное присваивание радикально меняет стиль решения ряда задач

Нахождение НОД

```
begin
  var (a, b) := (126,
  72);
  while b>0 do
    (a, b) :=      a mod b);
  (b,

  Println(a);
end.
```

```
18
```

Кролики Фибоначчи

```
begin
  var n :=
  15;
  var (a, b) := (1, 1);
  loop n do
  begin
    Print(a);
    (a,b) := (b, a +
    b);
  end;
end.
```

```
1 1 2 3 5 8 13 21 34 55 89     233
144                            377
```

Вывод

- Предпочтительные процедуры вывода – **Print** и **Println**. Они разделяют элементы вывода пробелами. Print удобно использовать в цикле для вывода серии значений
- Вывод по **интерполированной строке** позволяет обойтись одной строкой в случае сложного форматного вывода

Print

```
begin
  for var i:=1 to 10 do
    Print(i);
  Println;

  var (x,y) := (3,5);
  Println($'Сумма {x} и {y} равна {x+y}');
end.
```

```
1 2 3 4 5 6 7 8 9
10
Сумма 3 и 5 равна 8
```

Ввод

- Для ввода предпочтительно использовать функции ReadInteger, ReadReal и т.д.
- Они позволяют описать переменную и инициализировать её одной строкой
- Функции ReadInteger2, ReadReal2 и т.д. позволяют вводить сразу от 2 до 4 переменных одного типа

Пример. ReadReal, ReadReal2

```
begin
  var x := ReadReal('Введите
  x:');      var      y      :=
  ReadReal('Введите          y:');
  Println('Сумма =',x+y);
  var (a,b) :=                    катеты:')
  ReadReal2('Введите var c :=     ;
  Sqrt(a*a + b*b);
  Println('Гипотенуза =',c);
```

```
Введит  x: 1
е       y: 2
Введит  3
е       катеты: 3
Гипотенуза =
Сумма =
5
4
Введит
е
```

Цикл for

- Счётчик цикла for рекомендуется описывать непосредственно в заголовке цикла:
  `for var` `i`. В этом случае счётчик цикла for недоступен после цикла
- Если использовать старый синтаксис, то выдаётся **предупреждение** «Параметр цикла for в PascalABC.NET должен описываться в заголовке цикла»

Пример 1. Два последовательных цикла for

```
begin
  for var i:=0 to 9
  do
    Print(1+2*i);
  Println;          здесь
    // Переменная недоступна
   i for var i:=0 9 do
                 to
    Print(5+5*i);
end.
```

```
1 3 5 7 9 11 13 15 17 19
5 10 15 20 25 30 35 40 45
50
```

Расширенные операторы присваивания

- Расширенные операторы присваивания **+=**, **-=**, ***=** и **/=** встречаются во многих языках программирования
- «**+=**» читается как «увеличить на», а «***=**» читается как «увеличить в»
- a += 1 и a *= 2 воспринимается лучше чем a := a + 1 и a := a * 2

Пример 1. Сумма квадратов

```
begin
  var n := ReadInteger;
  var sum := 0;
  for var i:=1 to n do
    sum += i*i;
  Print($'Сумма квадратов первых {n} чисел ={sum}');
end.
```

```
10
Сумма квадратов первых 10 чисел = 385
```

Пример 2. 10!

```
begin
  var n := 10;
  var p := 1;
  for var i:=2 to n do
    p *= i;
  Print('10! =', p);
end.
```

```
10! = 3628800
```

Тип BigInteger

Тип BigInteger предназначен для вычислений с длинными целыми

Пример 1. 100!

```
begin
  var n := 100;
  var p: BigInteger := 1;
  for var i:=2 to n do
    p *= i;
  Write(n,'! = ', p);
end.
```

```
100! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
```

Пример 2. Тысячное число Фибоначчи

```
function Fib(n: integer): BigInteger;
begin
  var a,b: BigInteger;
  (a,b) := (1,1);
  loop n-1 do
    (a, b) := (b, a + b);
  Result := a
end;

begin
  Print(Fib(1000));
end.
```

```
43466557686937456435688527675040625802564660517371780402481729089536555417949051890403879840079255169295922593080322634775209689623239873322471161642996440906533187938298969649928516003704476137795166849228875
```

Возведение в степень

- Для возведения в степень используется операция **, которая реализована
  предельно эффективно

Пример. Вычисление 2 ** 1000

```
begin
  var b: BigInteger := 2;
  Println(b ** 1000);
end.
```

```
10715086071862673209484250490600018105614048117055336074437503883703510511
2
4936122493198378815695858127594672917553146825187145285692314043598457757
46
9857480393456777482423098542107460506237114187795418215304647498358194126
73
9876755916554394607706291457119647768654216766042983165262438683720566806
93
76
```

Case по строкам

В PascalABC.NET можно делать case по строкам. Это значительно удобнее старого стиля со вложенными if

Старый Паскаль

```
var Country: string;

begin
  read(Country);
  write('Столица:
  ');
  if Country = 'Россия' then
    writeln('Москва')
  else if Country = 'Франция' then
    writeln('Париж')
  else if Country = 'Италия' then
    writeln('Рим')
  else if Country = 'Германия' then
    writeln('Берлин')
  else writeln('Нет в базе данных');
end.
```

PascalABC.NET

```
begin
  var Country := ReadString;
  write('Столица: ');
  case Country of
    'Россия':   writeln('Москва');
    'Франция':  writeln('Париж');
    'Италия':   writeln('Рим');
    'Германия': writeln('Берлин');
    else writeln('Нет в базе
    данных');
  end;
end.
```

Result в функции

Для возвращения значения из функции следует использовать переменную
Result, а не устаревший синтаксис, связанный с присваиванием имени функции.
Переменная Result появилась в Delphi и используется во Free Pascal

```
function Fact(n: integer): integer;
begin
  Result := 1;
  for var i:=1 to n do
    Result *= i;
end;


begin
  var n := ReadInteger('Введите n:');
  Println('n!=',Fact(n));
end.
```

К оглавлению

Короткие определения функций и процедур

- В PascalABC.NET допускаются короткие определения для функций, задаваемых одним выражением. Тип возвращаемого значения выводится автоматически.
- Также допускаются короткие определения процедур, задаваемых одним оператором

Коллекция коротких определений подпрограмм

```
function ДлинаОкружности(r: real) := 2 * Pi * r;

function Гипотенуза(a,b: real) := Sqrt(a*a + b*b);

function Расстояние(x1,y1,x2,y2: real) := Гипотенуза(x2-x1,y2-y1);

function Минимум(a,b,c: real): real := Min(Min(a,b),c);

procedure Вывод(x: integer) := Println(x);

begin
  Println(ДлинаОкружности(1));
  Println(Гипотенуза(3,4),Расстояние
  (1,1,3,4)); Вывод(Минимум(5,3,8));
end.
```

К оглавлению

Упрощённый синтаксис модулей

В PascalABC.NET допускаются **модули** с упрощённым синтаксисом. В них можно описывать часто встречающиеся подпрограммы. Затем эти модули можно подключать к своим программам

Упрощённый синтаксис модуля

```
uses My;
function ДлинаОкружности(r: real) := 2 * Pi * r;

function Гипотенуза(a,b: real) := Sqrt(a*a + b*b);

function Расстояние(x1,y1,x2,y2: real) :=

  Гипотенуза(x2-x1,y2-y1);

function Минимум(a,b,c: real): real :=
  Min(Min(a,b),c);

procedure Вывод(x: integer) := Println(x);

end.
```

Основная программа

```
uses My;

begin
  Вывод(ДлинаОкружности(1));
  Вывод(Гипотенуза(3,4));
  Вывод(Минимум(5,3,8));
end.
```

Процедурные переменные

- Переменной можно присвоить действие. Такая переменная называется процедурной. Она хранит **отложенное действие**. Это действие можно вызвать, указав процедурную переменную вместо имени процедуры или функции
- Действие можно передать в подпрограмму как параметр. Вызов этого действия в этой подпрограмме называется **обратным вызовом** (callback)

Процедурная переменная

```
procedure Корова := Println('Му-у');
procedure Собака := Println('Гав!');
procedure Кошка := Println('Мяу!');

begin
  var Звук: procedure := Корова;
  Звук;
  Звук := Собака;
  Звук;
  Звук := Кошка + Корова * 2;
  Звук;
end.
```

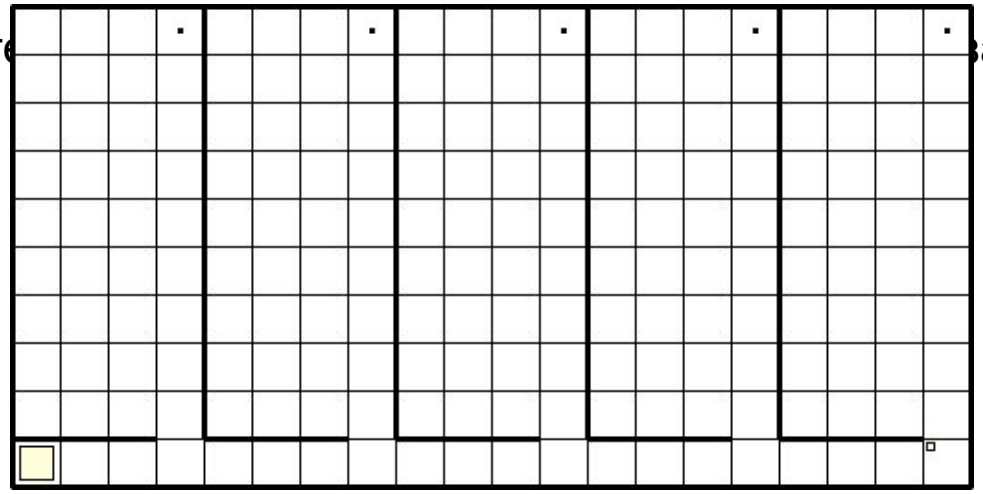
Callback-вызов

```
procedure Корова := Println('Му-у');
procedure Собака := Println('Гав!');
procedure Кошка := Println('Мяу!');

procedure ИздатьЗвуки(p1,p2: procedure);
begin
  p1; p2; // callback-вызовы
end;

begin
  ИздатьЗвуки(Корова, Собака);
  ИздатьЗвуки(Кошка, Корова);
end.
```

Операции + и * для процедур без параметров

Для процедур без параметров эффективно использовать операции + (последовательное выполнение) и *n (повторение n раз). Они позволяют получить **комбинированное действие**, которое можно вызвать как обычную процедуру без параметров.

Применение этой те задачи для Робота



```
uses Robot;
begin
  var d := Right*3 + Up*9 + Paint + Down*9;
  var d1 := (d + Right)*4 + d;
  d1;
end.
```

К оглавлению

Массивы – только динамические!

В PascalABC.NET в первую очередь следует использовать динамические массивы. Они содержат большое число методов

По ним можно выполнять цикл **foreach**

Циклы for и foreach по массивам

```
begin
  var a := Arr(1,3,5,7,9);
  foreach var x in a do
    Print(x);

  for var i:=0 to a.Length - 1 do
    a[i] += 1;
end.
```

Срезы массивов

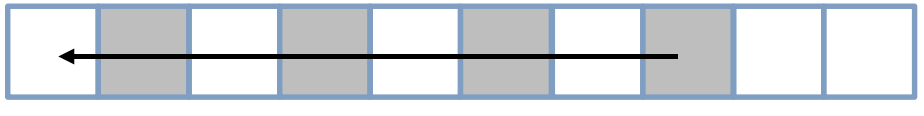
- Срез – это подмножество элементов массива в заданном диапазоне с заданным шагом
- Срезы имеют вид **a[from : to]** или **a[from : to : step]**
- Срезы с пропуском значений a[f :], a[: t], a[f : : s], a[: t : s], a[: : s]

a[2:5]

a[2:]

a[::2]

a[7::-2]



a[::-1]