PascalABC.NET
Простейшие новые возможности

- Описание переменных
- Множественное присваивание
- Ввод – вывод
- Циклы for и loop
- Расширенные операторы присваивания
- Тип BigInteger

Оглавление

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

Код

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

begin
  var a1: integer;
  var a2: integer := 5;
  var a3 := 555;            // a3 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;              // r1 получает тот же тип, что и r
  var m := Min(2,3);        // m получает тип, возвращаемый функцией Min
  var d := DateTime.Now;    // d получает тип, возвращаемый DateTime.Now
end.
```

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);




  (a,b) := (b,a);



  (a,b,c) := (b,c,a);




  var (имя, возраст) := ('Петя', 10);

end.
```

Эквивалентный код

```
begin
  var a,b,c: integer;

  a := 3;
  b := 5;
  c := 4;

  var t := a;
  a := b;
  b := t;

  var v := a;
  a := b;
  b := c;
  c := t;

  var имя := 'Петя';
  var возраст := 10;
end.
```

Вывод

- Предпочтительные процедуры вывода – Print и Println. Они разделяют элементы вывода пробелами. Print удобно использовать в цикле для вывода серии значений
- Процедуры Write и Writeln используются только для вывода без пробелов и для вывода по формату вида r:6:2 (для вещественных типов) или x:5 (для всех типов).
- Процедура WriteFormat осуществляет вывод по форматной строке

Print

```
begin
  var (x,y) := (1,2);
  Println(x,y);

  for var i:=1 to 10 do
    Print(i);
end.
```

```
1 2
1 2 3 4 5 6 7 8 9 10
```

Write, WriteFormat

```
begin
  var (a,b,c) := ('a','b','c');
  Writeln(a,b,c); // вывод без пробелов
  Writeln(Pi:6:2);
// 6 - ширина поля вывода
// 2 - количество знаков в дробной части

  var (x,y) := (3,5);
  WritelnFormat(
    'Сумма {0} и {1} равна {2}',x,y,x+y);
// При выводе на места {0} {1} {2} в форматной строке
// подставляются значения x, y, x+y
end.
```

```
abc
3.14
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 3 переменных
- Функции TryReadInteger, TryReadReal предназначены для безопасного ввода (если пользователь вводит не число, то они возвращают False)

Пример 1. ReadReal, ReadReal2

```
begin
  var x := ReadReal('Введите x:');
  var y := ReadReal('Введите y:');
  Println('Сумма =',x+y);

  var (a,b) := ReadReal2('Введите катеты:');
  var c := Sqrt(a*a + b*b);
  Println('Гипотенуза =',c);
end.
```

Пример 2. TryReadReal

```
begin
  var r: real;
  while not TryReadReal(r) do
    Println('Неверный ввод. Повторите');
// Первые 2 раза ошибочно вводим не числа
end.
```

```
1abc
Неверный ввод. Повторите
Hello
Неверный ввод. Повторите
3.14
```

Цикл loop

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

Пример 1. Геометрическая прогрессия

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

```
1 2 4 8 16 32 64 128 256 512 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;
  Print(s);
end.
```

```
2475
```

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;
  // Переменная i здесь недоступна
  for var i:=0 to 9 do
    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
```

Пример 2. Вывод таблицы квадратных корней

```
begin
  for var i:=1 to 9 do
    Println(i,Sqrt(i))
end.
```

```
1 1
2 1.4142135623731
3 1.73205080756888
4 2
5 2.23606797749979
6 2.44948974278318
7 2.64575131106459
8 2.82842712474619
9 3
```

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
```

Возведение в степень

- Для возведения в степень имеется также функция Power(x,y), а также более эффективные функции Sqr(x) и Sqrt(x) для возведения в квадрат и извлечения квадратного корня соответственно
- Для возведения в степень используется операция **, которая более эффективна чем функция Power при возведении в целую степень

Пример 1. Сумма квадратов

```
begin
  Println(Power(3,5), 3 ** 5);
  Println(Power(2,-10),2 ** -10);
  Println(Sqr(2.5), 2.5 ** 2);
  Println(Sqrt(2), 2 ** 0.5);

  var b: BigInteger := 2;
  Println(Power(b,50), b ** 50);
end.
```

```
243 243
0.0009765625 0.0009765625
6.25 6.25
1.4142135623731 1.4142135623731
1125899906842624 1125899906842624
```

Производительность при возведении в целую степень

```
begin
  var n := 20000000;
  var r := 0.0;
  var d := 100;
  for var i:=1 to n do
    r += (1.0-i/n) ** d;
  Println(r,MillisecondsDelta,'мс');

  r := 0.0;
  for var i:=1 to n do
    r += (1.0-i/n) ** real(d);
  Println(r,MillisecondsDelta,'мс');
// Производительность:
// при возведении в целую степень 100: 249 мс
// при возведении в вещественную степень 100: 1081 мс
end.
```

```
198019.301979065 249 мс
198019.301979065 1081 мс
```

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;
  Print(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
```

Самое важное

- В данной презентации содержатся основные конструкции, многие из которых присутствуют практически в каждой программе на PascalABC.NET
- Переменные практически всегда описываются с инициализацией и автовыводом типа: var p := 1
- Переменные преимущественно описываются внутри begin-end
- Множественное присваивание делает запись программы более короткой и понятной: (x,y) := (2,3)
- Классическая задача о перемене местами двух значений записывается максимально компактно с помощью множественного присваивания: (a,b) := (b,a)
- Расширенные операторы присваивания делают более понятной запись таких действий как «увеличить на 1» (a += 1) и «увеличить в 2 раза» (a *= 2)
- Тип BigInteger позволяет легко решать задачи, в которых возникают длинные целые: например, вычислять 100!
- Основная процедура вывода – Print, она разделяет элементы вывода пробелами
- Для вывода по формату используются либо форматы вывода x:5 и x:5:2 в операторе Write либо форматная строка и оператор WriteFormat
- Счётчик цикла for должен описываться в заголовке цикла: for var i
- Цикл loop используется, когда номер итерации цикла не важен
- Для ввода предпочтительно использовать функции ReadInteger, ReadReal и проч.
- Для ввода нескольких однотипных значений используются ReadInteger2 и проч.