PascalABC.NET
Процедуры, функции, лямбда-выражения

- Короткие определения функций и процедур
- Процедурные переменные
- Процедуры и функции обратного вызова
- Операции + и * для процедур без параметров
- Лямбда-выражения

Обновлено 19.01.18

Оглавление

Короткие определения функций и процедур

- В PascalABC.NET допускаются короткие определения для функций, задаваемых одним выражением
- Тип возвращаемого значения можно не указывать – он выводится автоматически
- Допускаются короткие определения процедур, задаваемых одним оператором

Код

```
function Куб(x: integer) := x*x*x;

function ПлощадьКруга(r: real) := Pi*r*r;

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

procedure DrawP(x,y: real) := Circle(x,y,3);
```

Эквивалентный код

```
function Куб(x: integer): integer;
begin
  Result := x*x*x;
end;

function ПлощадьКруга(r: real): real;
begin
  Result := Pi*r*r;
end;

function Hypot(a,b: real): real;
begin
  Result := Sqrt(a*a+b*b);
end;

procedure DrawP(x,y: real);
begin
  Circle(x,y,3);
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;
  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.
```

Процедурные переменные – функции

- Процедурной переменной можно присвоить функцию.
- Процедуру можно вызвать через процедурную переменную
- Действие можно передать в подпрограмму как параметр. Вызов этого действия в этой подпрограмме называется обратным вызовом (callback)

Вызов функции через процедурную переменную

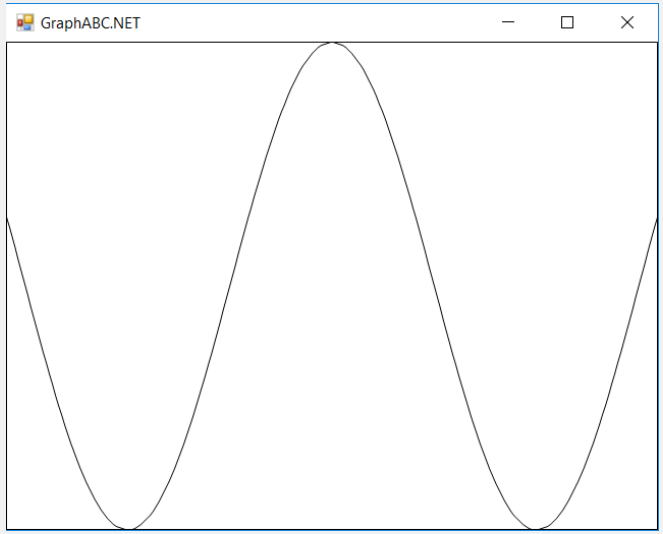
```
begin
  var a : real -> real := sin;
  Print(a(0));
  a := cos;
  Print(a(0));
end.
```

```
0 1
```

Передача функции как параметра

```
uses GraphABC;

begin
  Draw(cos);
end.
```



Операции с процедурами

- Процедуры без параметров можно складывать. p1 + p2 – это действие, состоящее в последовательном вызове процедур p1 и p2
- Процедуры без параметров можно умножать на число. p * n – это действие, состоящее в вызове процедуры p в цикле n раз
- Действия p1 + p2 и p * n можно вызывать только через процедурные переменные

Сложение процедур

```
procedure p := Write(1);
procedure q := Write(2);
procedure ln := Writeln;

begin
  var a1 := p + q + ln;
  a1;
  a1 := q + p + ln;
  a1;
end.
```

```
12
21
```

Умножение процедуры на число

```
procedure p := Write(1);
procedure q := Write(2);
procedure ln := Writeln;
procedure Run(p: procedure) := p;

begin
  a1 := p*10 + ln;
  a1;
  a1 := p*10 + q*10 + ln;
  a1;
  Run((p + q)*10 + ln);
end.
```

```
1111111111
11111111112222222222
12121212121212121212
```

Лямбда-выражения

- Лямбда-выражение – это анонимная функция, которая описывается в месте использования. Лямбда-выражения можно присваивать процедурным переменным и передавать как параметры.
- Вызов лямбда-выражения осуществляется через соответствующую процедурную переменную
- Лямбда-выражения – современный примитив программирования, используемый повсеместно

Код

```
begin
  var a := Arr(1,2,3,4,5,6,7,8,9);
  a.Where(x -> x mod 2 = 0) // Выбрать все чётные
   .Select(x -> x*x)        // возвести их в квадрат
   .Println;                // и вывести
end.
```

```
4 16 36 64
```

Эквивалентный код без лямбд

```
function Cond(x: integer): boolean;
begin
  Result := x mod 2 = 0;
end;

function Fun(x: integer): boolean;
begin
  Result := x*x+1;
end;

begin
  var a := Arr(1,2,3,4,5,6,7,8,9);
  a.Where(Cond)
   .Select(Fun).Println;
end.
```

```
4 16 36 64
```

Типы процедурных переменных

Типы процедурных переменных можно записывать в стиле определения лямбда-выражений:

- real->real – тип функции с параметром типа real, возвращающей real
- (real,real)->real – тип функции с двумя параметрами типа real, возвращающей real
- ()->integer – тип функции без параметров, возвращающей real
- (real,real)->() – тип процедуры с двумя параметрами типа real

Вызов функции через процедурную переменную

```
begin
  var dr: real->real := x->x+x;
  var ds: string->string := x->x+x;
  var add: (real,real)->real := (x,y)->x+y;
  var ran: ()->integer := ()->Random(1,9);
  var p: (real,real)->() := (x,y) ->
    WritelnFormat('{0}*{1}={2}',x,y,x*y);

  Println(dr(2),ds('abc'),add(2,3),ran());
  p(2,3);
end.
```

```
4 abcabc 5 4
2*3=6
```

Передача функции как параметра

```
procedure Transform(a: array of integer;
  f: integer->integer);
begin
  for var i:=0 to a.High do
    a[i] := f(a[i])
end;

begin
  var a := Arr(1,2,3,4,5); a.Println;
  Transform(a,x->x*x);
  a.Println;
end.
```

```
1 2 3 4 5
1 4 9 16 25
```

Самое важное

- Короткие определения процедур и функций значительно сокращают текст программы при наличии множества маленьких действий
- В переменных могут храниться не только данные, но и действия, в том числе и действия с параметрами
- Действия, хранящиеся в переменных, являются **отложенными**. Они вызываются через процедурную переменную: при вызове имя процедурной переменной пишется вместо имени подпрограммы
- Действия, хранящиеся в переменных, могут передаваться в другие подпрограммы в качестве параметров
- Процедуры без параметров можно складывать и умножать на число, после чего вызывать данное комбинированное действие через процедурную переменную. p1 + p2 означает p1; p2, а p * n означает loop n do p
- **Лямбда-выражения** используются в ситуации, когда для выполнения действия необходима одноразовая процедура или функция, которую неудобно специально описывать в разделе описаний
- Чтобы типы параметров и возвращаемого значения лямбда-выражения вывелись, лямбда-выражение надо присвоить процедурной переменной с известным типом
- Тип лямда-выражения можно записывать в виде со стрелкой, например : real -> real или (real, real) -> real
- Лямбда-выражения – современный примитив программирования, используемый во многих программах