Построение графиков дифференциальных уравнений

1 Практические рекомендации

1.1 Ссылочные параметры

В ряде случаев при написании программы возникает потребность изменить значение одного из параметров вызываемой процедуры или функции так, чтобы это изменение было «видно» в точке вызова этой процедуры или функции. Такая потребность может возникать при необходимости вернуть более одного значения из функции, при ограничениях на память и быстродействие, не позволяющих создавать новые объекты, а также при взаимодействии с операционной системой и сторонними библиотеками. В языке программирования Pascal для реализации такого поведения существуют **ссылочные параметры**.

Ссылочные параметры - параметры процедуры или функции, которые передаются *по ссылке*. При таком механизме передачи в памяти **не выделяется** место для хранения новой переменной; вместо этого для уже существующей памяти создаётся новое **имя**. Любые изменения, которые будут происходить с «новой» переменной, будут также отражаться и в «старой» - поскольку они будут использовать одну и ту же память.

В языке Pascal ссылочная переменная отмечается в списке параметров функции с помощью ключевого слова var, предшествующего её имени.

Приведём пример работы такой программы:

```
1  { Пример процедуры, получающей параметр по ссылке }
2  { Параметр a — ссылочный, b — "обычный", то есть  }
3  { будет передан "по значению". Изменения, произо— }
4  { шедшие с переменной a, отразятся на той перемен—}
5  { ной, которая была на первом месте при вызове    }
6  { процедуры.                                       }
7  procedure byReference(var a: Integer; b: Integer);
8  begin
9    a := a + b;
10   b := a;
11 end;
12 { Для демонстрации работы программы выведем на экран  }
13 { значения переменных, которые будем передавать в     }
14 { указанную процедуру.                                 }
15 begin
16   var x := 10;
17   var y := 20;
18   writeln(x, ' ', y);   { Выведет на печать: "10 20" }
19   byReference(x, y);
20   writeln(x, ' ', y);   { Выведет на печать: "30 20" }
21 end.
```

При вызове процедуры byReference первый параметр будет передан по ссылке, второй - по значению (то есть для переменной b будет выделена новая память, куда будет записано значение переменной y из основной программы). Поскольку переменные a процедуры byReference и x основной программы используют одну и ту же память, все изменения, происходящие с переменной a, перейдут в переменную x; в свою очередь, переменные b процедуры byReference и y основной программы будут использовать различную память и смогут изменяться независимо.

Аналогичным образом по ссылке можно передавать и массивы[1]. При этом передаваемые массивы могут быть не инициализированными (в случае с динамическими массивами), и память для них можно будет выделить прямо в вызванной функции. Приведём пример процедуры, производящей табулирование функции и записывающей результаты в два массива: один - для аргументов функции, другой - для соответствующих значений:

```
1  { Процедура табулирования функции          }
2  { Параметры:                               }
```

[1] Строго говоря, в PascalABC.Net массивы **всегда** передаются по ссылке; ключевое слово var следует указывать, если результат работы процедуры или функции будет записан в этот массив - это подсказка для программиста. Аналогично, если массив передаётся без ключевого слова var, то изменять его содержимое **недопустимо**.

```
3  {  a – левая граница отрезка табулирования  }
4  {  b – правая граница отрезка табулирования }
5  {  n – количество точек на отрезке          }
6  {       табулирования                       }
7  {  f – функция одной переменной, для которой}
8  {       строится таблица значений           }
9  {  x – массив, содержащий значения аргумента}
10 {  y – массов, содержащий значения функции  }
11 procedure tabulateRef(a, b: Real;
12                        n: Integer;
13                        f: Function(x: Real): Real;
14                        var x: array of Real;
15                        var y: array of Real);
16 begin
17   { Высчитываем шаг табулирования }
18   var h := (b - a) / (n - 1);
19   { Подготавливаем массивы x, y   }
20   SetLength(x, n);   SetLength(y, n);
21   { Производим непосредственно табулирование }
22   for var i := 0 to n - 1 do begin
23     x[i] := a + i * h;
24     y[i] := f(x[i]);
25   end;
26 end;
27 { Пример использования описанной процедуры }
28 begin
29   { Массивы для хранения значений табулирования }
30   var xf: array of Real;
31   var yf: array of Real;
32   { Табулирование функции sin(x)               }
33   tabulateRef(0, pi/2, 10, sin, xf, yf);
34   { Вывод результатов на печать. Поскольку     }
35   { размеры массивов xf и yf совпадают, можно  }
36   { использовать размер любого из них в качестве}
37   { пределов цикла                             }
38   for var i := 0 to xf.Length - 1 do begin
39     writeln(xf[i]:7:3, yf[i]:7:3);
40   end;
41 end.
```

В приведенном выше коде для массивов xf и yf основной программы память выделяется только в процедуре tabulateRef. При этом все изменения, произошедшие с массивами x и y, также происходят с массивами xf и yf.

Следует отметить, что использовать данную особенность языка следует весьма ограниченно. Процедуры и функции, передающие данные через ссылочные переменные, неочевидны в использовании и требуют дополнительного, обширного документирования. Часто наличие большого количества таких процедур и функций свидетельствует о недостаточной проработке архитектуры программы. В современном PascalABC.Net функции, возвращающие составные типы, предпочтительнее процедур, изменяющих значения массивов.

2 Задание

В качестве примера работы со ссылочными переменными предлагается написать программу, строящую графики решений дифференциальных уравнений. В качестве метода решения дифференциального уравнения следует использовать метод Эйлера.

Программа должна содержать процедуру odeEuler, принимающую в качестве параметров координаты начала и конца отрезка, на котором производится вычисление решения, количество точек для вычисления, функцию, по которой вычисляется значение производной, начальное решение, а также два массива для записи точек решения: по оси oX и по оси oY. Заголовок процедуры должен выглядеть так:

```
1 procedure odeEuler(a, b: Real; n: Integer;
```

```
2                       f: Function(x, y: Real): Real;
3                       y0: Real;
4                       var x: array of Real;
5                       var y: array of Real);
6  begin
7    SetLength(x, n);
8    SetLength(y, n);
9    { Далее - реализация метода Эйлера }
10 end;
```

Помимо этого, в программе должна присутствовать процедура plotArrays, на вход которой передаются два массива координат (по горизонтальной и вертикальной оси), и которая строит линию по значениям из этих массивов, автоматически подбирая параметры масштаба.

В основной программе должен происходить последовательный вызов процедуры odeEuler и plotArrays. Перед вызовом этих процедур следует запросить у пользователя параметры решения (координаты отрезка, начальное значение).

Уравнения использованы те же, что и в задаче по решению дифференциальных уравнений:

№	Уравнение
1	$y' = x^2$
2	$y' = y^2$
3	$y' = xy$
4	$y' = y - x$
5	$y' = sin(x) + y$
6	$y' = \frac{x^2-3x+9}{y}$