



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика»

## **Программирование в Delphi: подпрограммы**

Методические указания к лабораторной работе № 7  
по курсам «Информатика», «Алгоритмические языки  
и программирование»

Автор  
Е.Н. Ладоша, Д.С. Цымбалов, О.В. Яценко,  
А.П. Мул, Ю.Г. Зуева

Ростов-на-Дону, 2018

## Аннотация

Описаны приёмы и способы выделения относительно универсальных фрагментов алгоритма в подпрограммы (процедуры и функции) средствами Object Pascal. Целью работы ставится рационализация программирования в высокоуровневой среде разработки приложений Delphi. Предназначены для студентов всех специальностей факультета «Информатика и вычислительная техника».

## Автор

Доцент, к.т.н.  
Ладоша Е.Н.

Старший преподаватель кафедры  
«Электроника и электротехника»  
Цымбалов Д.С.

Доцент, к.ф.-м.н.  
Яценко О.В.

Старший преподаватель кафедры  
«Прикладная математика»  
Мул А.П.

Студент ДГТУ  
Зуева Ю.Г.



## Цель работы

Цель работы – изучить возможности и освоить приемы выделения часто используемых алгоритмических единиц программы в подпрограммы. Подробно рассмотрены особенности организации и вызова процедур, функция, а также основные способы передач параметров.

## Подпрограммы

В Object Pascal есть два вида подпрограмм: **процедуры и функции**.

Для процедур и функций общим является наличие списка фактических (формальных) параметров, приводимого в скобках после имени процедуры, или функции. Рассмотрим следующий пример. Процедура `Adder()` вычисляет сумму двух вещественных чисел `num1` и `num2`:

```
{Суммирование переменных num1 и num2 и сохранение результата в переменной sum}
procedure Adder(num1: Real; num2: Real; var sum: Real);
begin
    sum := num1 + num2 ;
end;
```

В этом коде переменные `num1`, `num2` и `sum` являются **формальными параметрами** процедуры `Adder()`. При вызове подпрограммы **формальным параметрам** присваиваются значения **фактических параметров**. Таким образом, в вызываемую подпрограмму передается информация из вызывающей подпрограммы.

Список формальных параметров, заданный в объявлении процедуры `Adder()` состоит из трех элементов. Это информирует компилятор о том, что в процедуру `Adder()` при вызове будут переданы три вещественных числа.

**Формальный параметр** – это переменная, которой при вызове подпрограммы присваивается значение соответствующего фактического параметра.

**Фактический параметр** – это значение, передаваемое в подпрограмму при ее вызове. Каждому **формальному параметру** соответствует **один фактический параметр**.

При составлении списка формальных параметров необходимо учитывать следующее.

1. Количество формальных параметров должно быть равно количеству фактических параметров.
2. Первый формальный параметр в списке соответствует первому фактическому, второй – второму и т.д.

3. Тип каждого фактического параметра должен совпадать с типом соответствующего ему формального параметра.
4. Имя фактического параметра никак не связано с именем соответствующего формального параметра.
5. Необходимо строго различать способы передачи данных – по ссылке, или по значению.

### Определение и использование подпрограмм

Процедуры и функции, не входящие в комплект поставки Delphi (т.е. не встроенные в Delphi), называются **пользовательскими подпрограммами**, потому что их должен определить программист (пользователь компилятора). Синтаксис объявления процедуры имеет следующий вид:

```
procedure имя_процедуры (формальный_параметр1: тип1;  
                        формальный_параметр2: тип2; . . .) ;  
[локальные_объявления; ]  
begin  
[операторы; ]  
end;
```

Чтобы вызвать процедуру, нужно задать ее имя и список фактических параметров. Синтаксис вызова процедуры имеет вид:

*имя\_процедуры (фактический\_параметр1, фактический\_параметр2, ...);*

Функции несколько отличаются от процедур. В функцию можно передавать любое количество параметров, однако она всегда возвращает одно значение в вызывающую подпрограмму. В то же время процедура не обязательно должна что-либо возвращать. *Если в вызывающую подпрограмму необходимо возвращать одно значение, то рекомендуется использовать функцию.*

Изменим процедуру Adder () на функцию:

```
{Функция Adder ()      возвращает сумму переменных num1 и  
num2}  
function Adder(num1: Real; num2: Real): Real;  
begin  
Adder := num1 + num2;  
end;
```

Синтаксис определения пользовательской функции имеет следующий вид:

```
function   имя_функции (формальный_параметр1 :      тип1;      ...) :
тип_функции;
[локальные_объявления; ]
begin
[операторы; ]
имя_функции := возвращаемое_значение;
end;
```

**Функция** – это подпрограмма, возвращающая одно значение, в вызывающую подпрограмму.

**Процедура** – это подпрограмма, которая не обязательно возвращает что-либо в вызывающую подпрограмму. Обычно процедуры предназначены для выполнения определенной задачи.

Обратите внимание: функция определяется с заданным типом *тип\_функции*. Этот тип имеет значение, возвращаемое из функции в вызывающую процедуру. Чтобы вернуть в вызывающую процедуру *возвращаемое\_значение*, оно должно быть присвоено имени *имя\_функции* в любом месте внутри блока функции (между ключевыми словами `begin` и `end`).

### Программирование: полезный совет

Внутри каждой пользовательской функции есть переменная `result`, которую можно использовать для возвращения значения вместо имени функции. Поэтому синтаксис функции может иметь следующий вид:

```
function   имя_функции (формальный_параметр1 :      тип1;      ...) :
тип_функции;
[локальные_объявления : ]
begin
[операторы; ]
result := возвращаемое_значение;
end;
```

Обратите внимание: переменную `result` не нужно объявлять, она всегда предполагается объявленной неявно. Как переменная `result`, так и переменная *имя\_функции* всегда содержат одно и то же значение. Когда выполнение функции прерывается, в вызывающую подпрограмму возвращается последнее значение, присвоенное или `result`, или имени функции.

Вызовом функции является записанное в вызывающей подпрограмме имя функции со списком фактических параметров в круглых скобках. В отличие от вызова процедуры вызов функции возвращает одно значение, следовательно, вызывающая подпрограмма должна с ним что-нибудь сделать (сохранить в пере-

менной, вывести на экран и т.д.). Синтаксис вызова функции с присвоением возвращаемого значения некоторой переменной имеет следующий вид:

*имя\_переменной* := *имя\_функции* (*фактический\_параметр1*,  
*фактический\_параметр2*, . . .) ;

### Программирование: полезный совет

Имеющие одинаковый тип и расположенные рядом формальные параметры могут быть сгруппированы. Например, определение функции `Adder()` можно записать следующим образом:

```
{Функция Adder() возвращает сумму переменных num1 и num2}  
function Adder(num1, num2: Real): Real;  
begin  
Adder := num1 + num2 ;  
end;
```

При группировании формальных параметров обязательно должна быть сохранена их последовательность.

### Передача параметров

**Передача по ссылке** – это передача в подпрограмму адреса (места расположения в памяти) фактического параметра. При этом подпрограмма работает непосредственно с передаваемым ей объектом.

**Передача по значению** – это передача в подпрограмму значения фактического параметра, т.е. копии объекта. При этом подпрограмма работает не с объектом, а с его копией. Передача по значению предохраняет объект от изменения подпрограммой. По умолчанию в Object Pascal параметры передаются по значению.

Параметры можно передавать в подпрограмму **по ссылке** или **по значению**. При передаче по ссылке в подпрограмму передается только информация о расположении переменной в памяти. Это позволяет подпрограмме получить доступ непосредственно к самой переменной. Следовательно, подпрограмма может изменить значение переменной.

В то же время при передаче по значению в подпрограмму передается значение переменной, т.е. подпрограмма получает доступ только к копии переменной. Изменить значение самой переменной подпрограмма не может. В Object Pascal по умолчанию параметры передаются по значению. Рассмотрим пример,

иллюстрирующие передачу параметра по значению:

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

var
  value: Integer;

procedure MyProcedure (number: Integer);
begin
  writeln('Executing MyProcedure.          number= ', number);
  number := 2*number;
  writeln('Finishing MyProcedure.        number= ', number);
end;

begin
  value := 5;
  writeln('Starting the test program.          number=
', value);
  MyProcedure(value);
  writeln('Finishing the program.          number=', value);
  readln;
end.
```

В этом фрагменте кода параметр `number` передается по значению. Программа выводит следующие строки:

```
Starting the test program.          number=5
Executing MyProcedure.              number=5
Finishing MyProcedure.              number=10
Finishing the program.              number=5
```

Обратите внимание: значение `value` не затронуто операциями в `MyProcedure`.

Модифицируем этот пример. Добавив перед объявление переменной `number`



ключевое слово `var`, означающее, что параметр `number` передается по ссылке.

```
procedure MyProcedure (var number: Integer);
```

При этом программа будет выводить следующие строки.

```
Starting the test program.      number=5
Executing MyProcedure.          number=5
Finishing MyProcedure.          number=10
Finishing the program.          number=10
```

Поскольку параметр `number` фактически является переменной `value`, процедура `MyProcedure` изменила значение `value`. В заголовке программы ключевое слово `var` должно предшествовать каждому параметру, передаваемому по ссылке.

### Программирование: полезный совет

В Object Pascal поддерживается также использование постоянных параметров (`const`). Постоянные параметры аналогичны передаваемым по значению за исключением того, что в теле процедуры или функции они не могут быть переприсвоены либо переданы другим подпрограммам по ссылке. Фактически постоянный параметр можно считать локальной именной константой.

### Передача массивов подпрограммам

В Object Pascal использование индексов в списке формальных параметров подпрограммы не допускается. Например, следующее объявление функции сгенерирует синтаксическую ошибку:

```
function Example(inArray: array[1..20] of Real): Real;
```

Синтаксическая ошибка возникает вследствие задания индексов параметра `inArray`. Напоминаем: чтобы обойти это ограничение, следует объявить пользовательский тип данных. Объявление этой же функции без синтаксической ошибки имеет вид

```
type
```

```
    RealArray20 = array[1..20] of Real;
```

```
function Example(inArray: RealArray20): Real;
```

В системе Delphi допускается использование открытых индексных параметров, т.е. массивов без указания индексов. Например, оператор

```
procedure OpenArrayEx(inArray: array of Integer);
```



объявляет процедуру, единственным формальным параметром которой является массив произвольной длины. Использование открытых индексных параметров существенно расширяет возможности подпрограмм. Синтаксис открытого индексного параметра напоминает синтаксис динамического массива, однако работают они по-разному. Для объявления динамического массива в качестве формального параметра для него должен быть определен пользовательский тип. Открытые индексные параметры подчиняются следующим правилам.

1. Массив, объявленный как открытый индексный параметр, всегда начинается с нуля, т.е. индекс первого элемента массива всегда равен 0.
2. Функции `Low()` и `High()` возвращают для такого массива значения 0 и *длина*-1. Функция `SizeOf()` возвращает длину фактического массива, переданного в подпрограмму.
3. В исходном коде можно обращаться только к элементам такого массива. Обращение к целому массиву запрещено.
4. Открытые индексные параметры не могут передаваться в процедуру `SetLength()`. Они могут передаваться в другие подпрограммы только как открытые индексные параметры или как нетипизированные параметры `var`.
5. Передаваемая в подпрограмму переменная базового типа открытого индексного параметра внутри подпрограммы полагается массивом единичной длины.

Массивы могут содержать огромное количество информации. При работе с большими массивами пользователю иногда легче вводить данные в файл, а не в поля ввода на пользовательском интерфейсе. Затем файл данных можно использовать как входной файл программы.

### Контрольные задания

1. Разработайте программу, которая приглашает пользователя ввести свое имя и фамилию, а затем в области просмотра выводит фамилию, запятую и имя (в этой последовательности). Выведите строку в области просмотра с помощью подпрограммы.
2. Разработайте программу, приглашающую пользователя ввести свое имя и фамилию и выводящую их буквами, переставленными в обратной последовательности. Инвертирование последовательности букв должно выполняться одной подпрограммой, а вывод результата – другой. Программа должна автоматически сделать первые буквы прописными. Например, Tom Smith должно быть выведено как Mot Ntims. (Встроенная подпрограмма изменения регистра буквы работает только для латинских букв. Не беспокойтесь о русских буквах – предполагайте, что пользователь вводит данные по-английски.)

3. Разработайте функцию `DecToHex ()`, преобразующие десятичные числа в шестнадцатеричные. Для проверки этих функций разработайте небольшую тестовую программу.

### Контрольные вопросы

1. Назовите два типа подпрограмм.
2. Дайте определение фактических и формальных параметров и опишите их соотношение.
3. Приведите синтаксис определения процедуры и функции. Выделите их отличия.
4. Приведите синтаксис вызовов процедуры и функции. Выделите их отличия.
5. В каких случаях использование функции предпочтительнее использования процедуры?
6. Что такое передача параметров по ссылке и по значению? Каким образом в программе можно задать вид передачи параметров?

### Задачи для самостоятельного выполнения

1. Найти площадь кругового кольца с заданным внешним  $R$  и внутренним  $r$  радиусами, используя процедуру вычисления площади круга.
2. Задан массив  $D(1..6)$ . Определить сумму каждой из следующих троек элементов массива:  $d1:d3$ ,  $d3:d5$ ,  $d2:d4$ . Составить процедуру вычисления суммы трех последовательно расположенных элементов массива  $D$  с номерами от  $k$  до  $m$ . Решение задачи потребует трехкратного обращения к процедуре.
3. На числовой оси расположены пять точек с координатами  $b1$ ,  $b2$ ,  $b3$ ,  $b4$ ,  $b5$  и точка  $A$  с координатой  $x$  ( $b1 \leq x \leq b5$ ). определить лежит ли точка  $A$  на одном из отрезков  $[b1, b2]$  или  $[b3, b4]$ . Составить процедуру, которая определяет, лежит ли точка  $A$  на отрезке  $[c,d]$  и, при выполнении условия, присваивает текстовой переменной, например,  $R$  значение "Да". В начале программы переменной  $R$  присвоить значение "Нет".
4. Составить процедуру вычисления функции:  $y=5x^2+10$  для значений  $x \in \{1..10\}$ .
5. Найти площадь фигуры которая образуется двумя квадратами. Внешний со стороной  $A$  и внутренним со стороной  $a$ , используя процедуру вычисления площади квадрата.
6. Задана матрица  $(3 \times 3)$ . Найти сумму любой строки (номер ввести с клавиатуры). Вычисление суммы оформить через процедуру. Все остальное – в программе.
7. В одномерном массиве заданы 7-мь чисел. Оформить в виде процедуры вычисление суммы  $[1,i]$ ,  $[2,i-1]$  и т.д. и использовать ее 3 раза. В программе все эти суммы сложить и вычесть средний элемент массива.

8. Составить процедуру вычисления функции:  $y=x^{1/2}+x$  для значений  $x=4, 16, 64, 256, 1024$ .
9. Дано три матрицы  $2*2$ . Для каждой вычислить сумму главной диагонали (это оформить в виде процедуры).
10. Задана матрица ( $3*3$ ). Найти сумму любого столбца (номер ввести с клавиатуры). Вычисление суммы оформить через процедуру. Все остальное – в программе.

### Список использованной литературы

1. Фаронов В.В. Delphi 3. Учебный курс. М.: «Нолидж», 1998. 400 с.
2. Галисеев Г.В. Программирование в среде Delphi 8 for .NET. М.: Издательский дом «Вильямс», 2004. 304 с.
3. Павловска Т.А. Паскаль. Программирование на языке высокого уровня. СПб.: Питер, 2003. 393 с.
4. Абрамов С.А. и др. Задачи по программированию. М.: Наука, 1988. 224 с.