



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра «Прикладная математика»

## **Программирование в Delphi: циклические структуры**

Методические указания к лабораторной работе № 4  
по курсам «Информатика», «Алгоритмические языки  
и программирование»

Автор  
Е.Н. Ладоса, Д.С. Цымбалов, О.В. Яценко,  
Е.В. Бочарова

Ростов-на-Дону, 2018



## Аннотация

Описаны способы и средства организации циклов в Object Pascal. Цель работы – выработать навыки реализации циклических алгоритмов в среде Delphi. Предназначены для студентов всех специальностей факультета «Информатика и вычислительная техника».

## Автор

Доцент, к.т.н.  
Ладоса Е.Н.

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

Доцент, к.ф.-м.н.  
Яценко О.В.

Студент ДГТУ  
Бочарова Е.В.



## Цель работы

Цели работы сводится к освоению **циклических структур**. Рассматриваются различные операторы цикла, включая `for`, `while` и `repeat`. Обсуждаются главные отличия между операторами цикла и ситуации, в которых они используются.

## Циклы

Иногда некоторую группу операторов согласно алгоритму нужно выполнить многократно. Для этого в программе используются операторы цикла. В языке Object Pascal программисту предоставлены три вида операторов цикла: `for`, `while` и `repeat`.

### Циклы `for`

Цикл `for` называется **детерминированным**. Это означает, что он может быть использован, только если количество итераций цикла можно определить до начала его выполнения. Синтаксис инкрементного цикла `for` имеет вид

```
for счетчик := начало to конец do begin
    [операторы; ]
end;
```

Переменная *счетчик* называется **счетчиком цикла**. Она должна быть локальной переменной любого **порядкового типа**. Выражения *начало* и *конец* определяют первое и последнее значение переменной *счетчик*. Тип выражений *начало* и *конец* должен быть **совместимым по присвоению** с типом переменной *счетчик*. Значение выражения *начало* должно быть меньше, или равно значению выражения *конец*, в противном случае операторы внутри цикла (в **теле цикла**) не будут выполнены ни разу. После выполнения цикла (после очередной итерации) инкрементный цикл `for` автоматически увеличивает значение переменной *счетчик* на единицу (или на одну порядковую позицию). Затем управление передается в **заголовок** цикла и *счетчик* сравнивается с выражением *конец*. Если *счетчик* меньше или равен выражению *конец*, то тело цикла выполняется повторно, в противном случае управление передается на оператор, следующий за оператором `for`, т.е. происходит выход из цикла.

Для целых числовых выражений *начало* и *конец* количество итераций оператора `for` равно

$$\begin{aligned} &(\text{конец} - \text{начало}) + 1, \text{ если } \text{начало} \leq \text{конец}; \\ &0, \text{ если } \text{начало} > \text{конец}. \end{aligned}$$

В следующем фрагменте кода инкрементный цикл `for` используется для вычисления суммы целых чисел от 1 до 100:

```
var
  counter: Integer;
  sum: Integer;
begin
  sum := 0;
  for counter := 1 to 100 do begin
    sum := sum + counter;
  end;
end;
```

Иногда счетчик цикла удобнее не увеличивать, а уменьшать. В этих случаях используется **декрементный** цикл `for`, синтаксис которого имеет вид

```
for счетчик := начало downto конец do begin
  [операторы; ]
end;
```

Как и раньше, переменная счетчик является счетчиком цикла, а выражения *начало* и *конец* определяют первое и последнее значения переменной *счетчик*. Однако есть и существенное отличие: в декрементном цикле значение *начало* должно быть больше или равно значению *конец*, иначе тело цикла не выполнится ни разу. После выполнения тела цикла *счетчик* автоматически уменьшается на единицу (или до предыдущей порядковой позиции). Если при этом *счетчик* больше или равен значению выражения *конец*, то выполняется тело цикла, в противном случае происходит выход из цикла.

Для целых числовых выражений *начало* и *конец* количество итераций декрементного цикла `for` равно

$$\begin{cases} \{\text{начало} - \text{конец}\} + 1, & \text{если } \text{начало} \geq \text{конец}; \\ 0, & \text{если } \text{начало} < \text{конец}; \end{cases}$$

Значение переменной счетчик увеличивается или уменьшается автоматически. Попытка изменить значение счетчик в теле цикла `for` вызовет сообщение об ошибке компиляции. Еще более опасная ошибка – попытка изменить в теле цикла значения выражений *начало* или *конец*. В этом случае компилятор не сообщит об ошибке, однако изменение на цикл не подействует, так как эти выражения были вычислены один раз в начале цикла и больше не вычисляются. Заметить такую ошибку очень трудно.

Цикл `for`, не содержащий в своем теле ни одного оператора, называется **пустым циклом**. В прошлом пустые циклы часто использовались для искусственной **задержки** выполнения программы, однако сейчас так делать нельзя. Современные компиляторы (в частности, компилятор Delphi) оптимизируют

выполнимый код. В режиме оптимизации компилятор увидит, что в цикле ничего не выполняется, и может пропустить цикл вообще. Как и операторы `if`, циклы `for` могут быть **вложенными**. Счетчик каждого вложенного цикла должен иметь уникальное имя.

### Циклы `while` и `repeat`

Циклы `while` и `repeat` являются **недетерминированными**. Это означает, что количество итераций не обязательно должно быть известно до начала выполнения цикла. Циклы `while` и `repeat` можно также использовать вместо `for` в качестве **детерминированных**. Таким образом, каждый цикл `for` можно заменить эквивалентным ему циклом `while` или `repeat`, но не наоборот. Используемые в Object Pascal циклы `while` и `repeat` имеют аналоги почти во всех языках высокого уровня.

Синтаксис цикла `while` имеет вид

```
while условие do begin  
    [операторы; ]  
end;
```

Синтаксис цикла `repeat` имеет вид

```
repeat  
    [операторы; ]  
until условие;
```

Выражение *условие* должно быть булева типа. Тело цикла `while` выполняется, если значение условия равно `True`. В то же время тело цикла `repeat` выполняется, если значение условия равно `False`. В этих циклах (в отличие от `for`) в теле цикла могут явно изменяться переменные, входящие в выражение *условие*, которое в этом случае называется **управляющим выражением**. Если *условие* в теле цикла не изменяется, то `while` или `repeat` превращается в **бесконечный цикл**.

Обратите внимание: в приведенном выше синтаксисе ключевые слова `begin` и `end` ограничивают тело цикла `while`, но не цикла `repeat`. Тело цикла `repeat` составляют все операторы, расположенные между ключевыми словами `repeat` и `until`.

Существует еще одно важное отличие между циклами `while` и `repeat`: тело цикла `repeat` всегда выполняется, как минимум, один раз, в то время как тело цикла `while` может не выполняться не разу.

Поэтому цикл `repeat` легко преобразовать в цикл `while`, однако обратное преобразование – более сложная задача. Тот фактор, что тело цикла `while` может оказаться не выполненным ни разу, придется учитывать с помощью до-

полнительного условия `if`, проверяющего условие в начале цикла. Естественно, такая проверка избыточна: программист вынужден дважды вводить один и тот же код условия. Поэтому данный фактор является основным условием выбора между циклами `while` и `repeat`: если тело цикла всегда должно быть выполнено хотя бы один раз, то предпочтителен цикл `repeat`, в противном случае более удобен цикл `while`.

В качестве примера использования циклов `while` и `repeat` запишем с их помощью код вычисления суммы целых чисел от 1 до 100:

```
{Использование цикла while}
sum := 0;
count := 1;
while (count <= 100) do begin
    sum := sum + count;
    count := count + 1;
end;
{Использование цикла repeat}
sum := 0;
count := 1;
repeat
    sum := sum + count;
    count := count + 1;
until (count > 100);
```

В этих примерах переменная `sum` является **накопителем**. В ней суммируются (накапливаются) значения, образующие окончательный ответ. Обратите внимание: эти два кода очень похожи. Фактически они отличаются только условиями прерывания циклов. Для решения данной задачи одинаково пригоден любой цикл – `for`, `while` или `repeat`. В общем случае выбор вида цикла определяется потребностями оптимизации кода и стиля программирования.

В циклах `while` и `repeat` часто используются **флажки**. Флажок представляет собой переменную, принимающую разные значения при удовлетворении или неудовлетворении некоторого условия либо изменяющую свое значение при наступлении некоторого события. Например, в следующем фрагменте кода флажком служит переменная `correct`. Цикл выполняется, пока пользователь не введет ответ Yes или No.

```
program Project2;

{$APPTYPE CONSOLE}

uses
    SysUtils;

var
    answer:    String;
    correct:   Boolean;
begin
    repeat
```



```
writeln('Enter answer Yes or No');  
readln(answer);  
answer := Trim(Uppercase(answer));  
correct := (answer = 'YES') or (answer = 'NO');  
until correct;  
end.
```

Как вы помните, счетчик цикла `for` должен быть переменной любого порядкового типа. Допустим, нужно преобразовать цикл `for` с нецелочисленным счетчиком в эквивалентный ему цикл `while` или `repeat`. Тогда в коде необходимо предусмотреть увеличение или уменьшение значения счетчика до следующего или предыдущего порядкового значения. Эта задача решается с помощью встроенных порядковых подпрограмм, перечисленных в табл. 1.

Таблица 1

Встроенные порядковые подпрограммы Object Pascal

Функция	Назначение
<b>Odd</b> (X: Integer): Boolean;	Возвращает True, если аргумент является нечетным числом
<b>Ord</b> (X: Char): Byte;	Возвращает порядковое значение выражения порядкового типа
<b>Pred</b> (X: Integer): Integer;	Возвращает предыдущее порядковое значение аргумента
<b>Succ</b> (X: Integer): Integer;	Возвращает следующее порядковое значение аргумента
Процедура	Назначение
<b>Dec</b> (var X: Integer); <b>Dec</b> (var X: Integer; N: Integer);	Уменьшает порядковую переменную на 1 или на n
<b>Inc</b> (var X: Integer); <b>Inc</b> (var X: Integer; N: Integer);	Увеличивает порядковую переменную на 1 или на n

Как показано в табл. 1, процедура `Inc()` увеличивает порядковое значение на заданную целую величину. По умолчанию, если эта величина не указана, то порядковая переменная увеличивается на 1. Таким образом, вызов `Inc(x)` эквивалентен оператору `x := x + 1`, а `Inc(x, n)` – оператору `x := x + n`. Аналогично этому работает и процедура `Dec()`. Она уменьшает порядковое значение на указанную целую величину. Характерной особенностью процедур `Inc()` и `Dec()` является то, что они генерируют оптимизированный код, поэтому они особенно полезны в циклах.

Следующий фрагмент кода записывает в строку `alphabet` буквы английского алфавита в нижнем регистре. В приведенном фрагменте используется цикл `while`. В качестве упражнения напишите эквивалентный код с циклом `repeat`.

```
var  
letter: Char;  
alphabet: String;
```



```
begin
  alphabet := ' ';
  letter := 'a';
  while (letter <= 'z') do begin
    alphabet := alphabet + letter;
    letter := Succ(letter);
  end;
end;
```

## Процедуры BREAK и CONTINUE

В Object Pascale в циклах можно использовать стандартные процедуры BREAK и CONTINUE. Процедура BREAK позволяет выйти из цикла, не дожидаясь выполнения условия выхода. Процедура CONTINUE позволяет начать новую итерацию цикла, даже если предыдущая еще не завершена.

## Контрольные задания

1. Организовать ввод с клавиатуры последовательности натуральных чисел. Признаком конца ввода является ввод числа 0. В процессе ввода подсчитать число введенных чисел и определить их среднее значение.
2. Изменить программные реализации решения Задачи 1, дополнив их определением наибольшего и наименьшего из введенных чисел.
3. Написать программу, которая проверяет, является ли введенное пользователем число простым. (Простым называется число, которое не имеет делителей кроме 1 и самого себя).
4. Найдите наибольший общий делитель двух чисел введенных с клавиатуры.
5. Составьте таблицу значений функции  $y = e^{\sin(x)} \cos(x)$  на отрезке  $[0; \pi]$ , шаг изменения аргумента 0,1.

## Контрольные вопросы

1. Опишите синтаксис циклов for, while и repeat и выполняемые ими операции.
2. Существует ли для каждого цикла for эквивалентный ему цикл while или repeat? Если нет, то почему?
3. Существует ли для каждого цикла while или repeat эквивалентный ему цикл for? Если нет, то почему?
4. Можно ли каждый цикл while заменить эквивалентным ему циклом repeat? Если нет, то почему?
5. Можно ли каждый цикл repeat заменить эквивалентным ему циклом



while? Если нет, то почему?

6. Что такое счетчик? Что такое флажок?
7. Назовите порядковые подпрограммы и опишите их назначение.
8. Какие циклы называются вложенными?
9. Можно ли войти в тело цикла `for`, минуя его заголовок?
10. Можно ли войти в тело цикла `while`, минуя его заголовок?
11. Можно ли войти в тело цикла `repeat`, минуя его заголовок?
12. Могут ли внешний и внутренний циклы быть циклами разных видов?
13. Могут ли внешний и внутренний циклы быть циклами одного вида?
14. Может ли вещественная переменная быть параметром цикла `for`?
15. Может ли булевская переменная быть параметром цикла `for`?

### Задачи для самостоятельного выполнения

1. Для введенного с клавиатуры целого числа выяснить, что больше  $n!$  или  $e^n$ . ( $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$ ).
2. Найти любое трёхзначное число, кратное заданному  $P$  и не равное ему.
3. С клавиатуры вводится целочисленная последовательность. Ввод продолжается до ввода 0. Найти сумму четных элементов и произведение нечетных.
4. С клавиатуры вводится целочисленная последовательность. Ввод продолжается до того как сумма всех введенных чисел превысит (станет больше 100). Вывести сумму на экран.
- Вычислить  $P = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 4 \cdot 5 \cdot 6 + \dots + N \cdot (N+1) \cdot \dots \cdot 2N$ .
5. Пусть дано натуральное число  $n$ . Выбросите из записи этого числа цифры 3 и 7, оставив прежним порядок остальных цифр. Например, из числа 3171507377 должно получиться 1150.
6. Пусть дано натуральное число  $n$ . Получите наименьшее число вида  $2^k$ , превосходящее  $n$ .
7. Составьте программу для нахождения всех автоморфных чисел в отрезке  $[n, m]$ . Автоморфным называется целое число, которое равно последним числам своего квадрата. Например:  $5^2=25$ ,  $6^2=36$ ,  $25^2=625$ .
8. Вычислите наименьший общий делитель двух натуральных чисел.
9. Каждая бактерия делится на две в течении одной минуты. В начальный момент времени имеется одна бактерия. Составьте программу для расчета количества бактерий через заданное целое количество минут.
10. Пусть интервал  $(a, b)$  разбит точками на  $n$  равных частей. В каждой точке вычисляется значение функции  $\frac{x^2 - 3x + 2}{\sqrt{2x^3 - 1}}$ . Найти наибольшее и наимень-

шее значение в этих точках.

11. Вычислите определенный интеграл  $\int_0^{0.5} 4 \cos^2 x dx$ .

12. Вычислите определенный интеграл  $\int_0^9 \frac{x+1}{\sqrt{x}} dx$ .

13. Вычислите определенный интеграл  $\int_0^{\pi/3} \frac{dx}{\cos^2 x}$ .

### 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.