



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра «Прикладная математика»

Программирование в Delphi: рекурсия

Методические указания к лабораторной работе № 8
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 реализации циклических алгоритмов (рекурсии) в 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 приёмы составления **рекурсивных алгоритмов** с последующей реализацией в среде программирования Delphi. Рассмотрены авто- и перекрестная рекурсии.

Рекурсивные подпрограммы

Рекурсивной называется подпрограмма, в которой содержится обращение к самой себе. Такая рекурсия называется **прямой (авторекурсией)**. Есть также **косвенная (перекрестная)** рекурсия, когда две или более подпрограмм вызывают друг друга.

При обращении подпрограммы к самой себе происходит то же самое, что и при обращении к любой другой функции или процедуре: в стек записывается адрес возврата, резервируется место под локальные переменные, происходит передача параметров, после чего управление передается первому исполняемому оператору подпрограммы. При повторном вызове этот процесс повторяется. Ясно, что для завершения вычислений каждая рекурсивная подпрограмма должна содержать хотя бы одну *нерекурсивную ветвь алгоритма*, заканчивающуюся возвратом в вызывающую программу.

При завершении подпрограммы область ее локальных переменных освобождается, а управление передается на оператор, следующий за рекурсивным вызовом. Простой пример рекурсивной функции — вычисление факториала (это не означает, что факториал следует вычислять именно так). Чтобы получить значение факториала числа n , требуется умножить на n факториал числа $(n-1)$. Известно также, что $0! = 1$ и $1! = 1$.

```
function fact(n : byte) : longint;  
begin  
    if (n = 0) or (n = 1) then fact := 1  
    else fact := n * fact(n - 1);  
end;
```

Рассмотрим, что происходит при вызове этой функции при $n = 3$. В стеке отводится место под параметр n , ему присваивается значение 3, и начинается выполнение функции. Условие в операторе `if` ложно, поэтому управление передается на ветвь `else`. Для вычисления выражения $n * \text{fact}(n-1)$ требуется повторно вызвать функцию `fact`. Для этого в стеке отводится новое место под параметр n , ему присваивается значение 2, и выполнение функции начинается сначала. В третий раз функция вызывается со значением параметра, равным 1, и вот тут-то становится истинным выражение $(n = 0) \text{ or } (n = 1)$, поэтому происходит

возврат из подпрограммы в точку вызова, то есть на выражение $n * \text{fact}(n-1)$ для $n=2$. Результат выражения присваивается имени функции и передается в точку ее вызова, то есть в то же выражение, только теперь происходит обращение к параметру n , равному 3.

Понимание механизма рекурсии помогает осознать ее достоинства, недостатки и область применения.

Рекурсивные подпрограммы чаще всего применяют для компактной записи рекурсивных алгоритмов, а также для работы со структурами данных, описанными рекурсивно, например с двоичными деревьями. Любую рекурсивную функцию можно реализовать без применения рекурсии: для этого программист должен сам обеспечить распределение памяти под необходимое количество копий параметров.

Достоинством рекурсии является компактная запись. К недостаткам относятся расход времени и памяти на повторные вызовы функции и передачу ей параметров, а главное, опасность переполнения стека.

При отладке рекурсивных алгоритмов полезно отслеживать глубину рекурсии либо визуально, вставив оператор вывода в начало подпрограммы, либо с помощью типизированной константы, которая увеличивается на единицу при каждом вызове подпрограммы, например:

```
function fact(n : byte): longint;  
const num: word = 0;  
begin  
  inc(num);  
  writeln(num);    { отладочная печать }  
  if (n = 0) or (n = 1) then fact := 1  
  else fact := n * fact(n - 1);  
end;
```

Напомню, что локальная типизированная константа, в отличие от локальной переменной, размещается в сегменте данных и сохраняет свое значение между вызовами подпрограммы.

В случае переполнения стека программа завершится с соответствующим сообщением об ошибке. Проверка переполнения стека задается ключом компилятора `{SS+}`. По умолчанию он включен.

Задачи

1. Напишите рекурсивную функцию для нахождения биномиальных коэффициентов (для заданного $M \geq i \geq j > 0$ вычислите все C_i^j):

$$C_m^n = \begin{cases} 1, & \text{при } m=0, n>0 \text{ или } m=n \geq 0 \\ 0, & \text{при } m>n>0, \\ C_{m-1}^{n-1} + C_m^{n-1}, & \text{иначе} \end{cases}$$

2. Напишите рекурсивную функцию, которая по заданным вещественному x и целому n вычисляет величину x^n согласно формуле:

$$x^n = \begin{cases} 1, & n = 0 \\ \frac{1}{x^{|n|}}, & n < 0 \\ x(x^{n-1}), & n > 0 \end{cases}$$

Теоретические вопросы

1. Как называются процедуры или функции, которые вызывают сами себя?
2. Для каких целей создаются рекурсивные алгоритмы?
3. Что называется стеком?
4. Верно ли, что значения всех локальных переменных при очередном вызове рекурсивной процедуры или функции помещаются в стек?
5. В какой последовательности происходит заполнение стека и выбор элементов из стека?
6. Всегда ли в рекурсивном алгоритме должно присутствовать условие выхода из рекурсии?
7. Что произойдет, если рекурсивный алгоритм будет вызывать сам себя «бесконечное» число раз?
8. Верно ли, что решение задачи, реализуемое рекурсивным алгоритмом, можно выразить нерекурсивным алгоритмом?

Задания для самостоятельной работы

1. По заданным границам интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \sin^n x dx = \begin{cases} -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x dx, & n > 2 \\ \frac{x}{2} - \frac{1}{4} \sin 2x, & n = 2 \\ -\cos x, & n = 1 \end{cases}$$

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.