Практическая работа №2. Рекурсия

Цель работы: познакомиться с одним из эффективных способов решения сложных задач – рекурсией.

Общие сведения:

Очень часто, разрабатывая программу, удается свести исходную задачу к более простым. Среди этих задач может оказаться и первоначальная, но в упрощенной форме. Например, вычисление функции F(n) может потребовать вычисления F(n-1) и еще каких-то операций. Иными словами, частью алгоритма вычисления функции будет вычисление этой же функции.

Алгоритм называется рекурсивным, если он прямо или косвенно обращается к самому себе. Часто в основе такого алгоритма лежит рекурсивное определение какого-то понятия. Например, о факториале числа N можно сказать, что N! = N*(N – 1)!, если N > 0 и N! = 1 если N = 0. Это – рекурсивное определение.

Вот еще одно рекурсивное определение.

1. 3 коровы – это стадо коров.
2. Стадо из n коров – это стадо из n – 1 коровы и еще одна корова.

Попробуем применить это определение для проверки, является ли стадом группа из пяти коров (обозначим ее К5). Объект К5 не удовлетворяет первому пункту определения, поскольку пять коров – это не три коровы. Согласно второму пункту К5 – стадо, если там есть одна корова, а остальная часть К5, назовем ее К4, – тоже стадо коров. Решение относительно объекта К5 откладывается, пока не будет принято решение относительно К4. Объект К4 снова не подходит под первый пункт, а второй пункт гласит, что К4 – стадо, если объект К3, полученный из К4 путем отделения одной коровы, тоже стадо. Решение о К4 тоже откладывается. Наконец, объект К3 удовлетворяет первому пункту определения, и мы можем смело утверждать, что К3– стадо коров. Теперь и о К4 можно утверждать, что это стадо, а значит, и К5 является стадом коров.

Любое рекурсивное определение состоит из двух частей. Эти части принято называть базовой и рекурсивной частями. Базовая часть является нерекурсивной и задает определение для некоторой фиксированной части объектов. Рекурсивная часть определяет понятие через него же и записывается так, чтобы при цепочке повторных применений она редуцировалась бы к базе.

Примеры

1. Задача. *Написать рекурсивную программу поиска минимального элемента массива.*

Решение. Опишем функцию Pmin, которая определяет минимум среди первых n элементов массива a. Параметрами этой функции являются количество элементов в рассматриваемой части массива - n и значение последнего элемента этой части – a[n]. При этом если n>2, то результатом является минимальное из двух чисел – a[n] и минимального числа из первых (n-1) элементов массива. В этом заключается рекурсивный вызов. Если же n=2, то

результатом является минимальное из первых двух элементов массива. Чтобы найти минимум всех элементов массива, нужно обратиться к функции Pmin, указав в качестве параметров значение размерности массива и значение последнего его элемента. Минимальное из двух чисел определяется с помощью функции Min, параметрами которой являются эти числа.

```
Program Example _1;
Const n=10;
Type MyArray=Array[1..n] of Integer;
Const a : MyArray = (4,2, -1,5,2,9,4,8,5,3);
Function Min (a, b : Integer) : Integer;
Begin
        if a>b then Min := b else Min:=a;
End;
Function Pmin(n, b : Integer) : Integer;
Begin
        if n = 2 then Pmin := Min(n,a[1]) else Pmin := Min(a[n], Pmin(n-1,a[n]));
End;
BEGIN
        Writeln('Минимальный элемент массива  - ', Pmin(n,a[n]));
END.
```

2. Задача. *Ханойские башни.*

Имеется три стержня A, B, C. На стержень A нанизано n дисков радиуса 1, 2,..., n таким образом, что диск радиуса i является i-м сверху. Требуется переместить все диски на стержень B, сохраняя их порядок расположения (диск с большим радиусом находится ниже). За один раз можно перемещать только один диск с любого стержня на любой другой стержень. При этом должно выполняться следующее условие: на каждом стержне ни в какой момент времени никакой диск не может находиться выше диска с меньшим радиусом.

Решение. Предположим, что мы умеем перекладывать пирамиду из (n-1) диска. Рассмотрим пирамиду из n дисков. Переместим первые (n-1) дисков на стержень C (это мы умеем). Затем перенесем последний n-й диск со стержня A на стержень B. Далее перенесем пирамиду из (n-1) диска со стержня C на стержень B. Так как n-й диск самый большой, то условие задачи не будет нарушено. Таким образом, вся пирамида будет на стержне B. Аналогичным образом можно перенести n – 2, n – 3 и т. д. дисков. Когда n=1, осуществить перенос очень просто: непосредственно с первого стержня на второй. При этом для решения задачи будет достаточно 2n - 1 перекладываний.

```
Program Example_2;
Const k = 3;
Var a,b,c : Char;
Procedure Disk(n : Integer; a, b, c: Char);
Begin
if n>0 then
    begin
      Disk(n-1,a,c,b);
Writeln('Диск ',n, ' c ', a,'->', b);
```

```
      Disk(n-1,c,b,a);
    end;
End;
BEGIN
a := 'A'; b := 'B'; c := 'C';
  Disk(k,a,b,c);
ReadLn;
END.
```

Контрольные вопросы

1. На чем основан рекурсивный метод программирования?
2. В чем разница между «циклическим» и «рекурсивным» способами определения? Какой элемент является обязательным в рекурсивном определении?
3. Что такое «фрейм активации»?
4. К каким последствиям приводит «рекурсивное зацикливание»?
5. Какое условие должно обязательно присутствовать в любой рекурсивной процедуре?
6. Что такое явная и косвенная рекурсии?
7. Дайте рекурсивное определение целой степени числа N.

Варианты заданий:

1. Найдите первые N чисел Фибоначчи двумя способами: с помощью рекурсии и с помощью итерации. Сравните эффективность алгоритмов.

2. Напишите традиционную функцию умножения двух чисел и функцию, использующую только операцию сложения. Сравните эффективность алгоритмов.

3. Напишите метод, находящий максимальное из двух чисел, не используя операторы if-else или любые другие операторы сравнения. Оцените сложность алгоритма.

4. Напишите функцию суммирования двух целых чисел без использования «+» и других арифметических операторов. Оцените сложность алгоритма.

5. Вычислите несколько значений функции Аккермана для неотрицательных чисел m и n. Оцените сложность.

$$A(n,m) = \begin{cases} m+1, & n = 0 \\ A(n-1,1), & n \neq 0, m = 0 \\ A(n-1, A(n,m-1)), & n > 0, m \geq 0 \end{cases}$$

6. Вычислите произведение элементов одномерного массива двумя способами: с помощью рекурсии и с помощью итерации. Оцените сложность алгоритма.

7. Вычислите, используя рекурсию, выражение:

$$\sqrt{6+2\sqrt{7+3\sqrt{8+4\sqrt{9+\ldots}}}}$$