Лабораторная работа № 32-33

«Разработка, оценка сложности и оформление рекурсивного алгоритма»

Цель работы:
Изучить оформление и оценку сложности рекурсивного алгоритма.

Теория

Простая рекурсия

Напомним, что рекурсивными процедурами называются процедуры, которые вызывают сами себя. Их сложность определить довольно тяжело. Сложность этих алгоритмов зависит не только от сложности внутренних циклов, но и от количества итераций рекурсии. Рекурсивная процедура может выглядеть достаточно простой, но она может серьёзно усложнить программу, многократно вызывая себя.

Рассмотрим рекурсивную реализацию вычисления факториала:

```
function Factorial(n: Word): integer;
begin
if n > 1 then
Factorial:=n*Factorial(n-1)
else
Factorial:=1;
end;
```

Эта процедура выполняется N раз, таким образом, вычислительная сложность этого алгоритма равна O(N).

Многократная рекурсия

Рекурсивный алгоритм, который вызывает себя несколько раз, называется многократной рекурсией. Такие процедуры гораздо сложнее анализировать, кроме того, они могут сделать алгоритм гораздо сложнее.

Рассмотрим такую процедуру:

```
procedure DoubleRecursive(N: integer);
begin
if N>0 then
begin
DoubleRecursive(N-1);
DoubleRecursive(N-1);
end;
end;
```

Поскольку процедура вызывается дважды, можно было бы предположить, что её рабочий цикл будет равен O(2N)=O(N). Но на самом деле ситуация гораздо сложнее. Если внимательно исследовать этот алгоритм, то станет очевидно, что его сложность равна O(2^(N+1)-1)=O(2^N). Всегда надо помнить, что анализ сложности рекурсивных алгоритмов весьма нетривиальная задача.

Честно говоря, этого примера недостаточно, чтобы убедить вас использовать рекурсивный метод. Если можете написать циклический алгоритм — то всегда используйте только его. Однако есть забавная задача, которая решается только через рекурсию (мне не попадалось ее решение через цикл для произвольного n) — это Ханойские башни:

«Требуется за минимальное число операций переложить n алмазных колец разного диаметра пирамидки с 1-й на 3-ю через 2-ю иглу. Запрещено перекладывать более одного кольца за одну операцию и помещать кольцо большего диаметра над меньшим (в целях сохранности). В исходном положении — все кольца на игле 1, в конечном — на игле 3 имеют форму пирамиды» :

```
using System;

namespace HanoiTowers
{

    class Program
    {

        static void Main(string[] args)
        {
            Console.Write("n="); // число алмазных колец
            int n = Convert.ToInt32(Console.ReadLine());
            hanoi(n, '1', '2', '3'); // Вызов. Иглы 1,2,3
            Console.ReadKey();
        }

        // Рекурсивная функция

        static void hanoi(int n, char a, char b, char c )
        {
            if (n < 2)
                Console.WriteLine(a.ToString() + "->" + c.ToString());
            else
            {
                hanoi(n - 1, a, c, b);
                Console.WriteLine(a.ToString() + "->" + c.ToString());
                hanoi(n - 1, b, a, c);
            }
        }
```

```
    }
}
```

Минимальное число операций равно 2n-1. Попробуйте придумать более быстрый алгоритм, не изменяя условия задачи. При n=30 мы имеем более 1 миллиарда операций.

Ход работы:

1. Напишите функции вычисления площадей следующих фигур: квадрат, прямоугольник, треугольник (значения сторон выберите сами)
2. Создайте рекурсивную функцию, вычисляющую факториал переданного числа

```
static void Main(string[] args)
{
    onChoice();

    void onChoice()
    {
        Console.WriteLine("Выбирете действие:");
        Console.WriteLine("0: поиск площади квадрата");
        Console.WriteLine("1: поиск площади прямоугольника");
        Console.WriteLine("2: поиск площади треугольника");
        Console.WriteLine("3: вычисление факториала числа");
        Console.WriteLine("4: выход");
        Console.Write("> ");
        try
        {
            int choice = Convert.ToInt32(Console.ReadLine());

            switch (choice)
            {
                case 0:
                    Console.WriteLine("\nплощадь квадрата: " + getSquareArea() + "\n

                    break;
                case 1:
                    Console.WriteLine("\nплощадь прямоугольника: " + getRectangleAre
                    break;
                case 2:
                    Console.WriteLine("\nплощадь треугольника: " + getTriangleArea()
                    break;
                case 3:
                    Console.WriteLine("\nфактриал = " + getFactorial() + "\n");
                    break;
                case 4:
                    System.Environment.Exit(0);
                    break;
```

```
                default:
                    Console.WriteLine(); onChoice();
                    break;
            }
        }
        catch
        {
            Console.WriteLine("\nВведите корректное значение\n");
        }
    }

    int getSquareArea()
    {
        int a = input("a");

        return a * a;
    }

    int getRectangleArea()
    {
        int a = input("a");

        int b = input("b");

        return a * b;
    }

    double getTriangleArea()
    {
        int a = input("a");
        int b = input("b");
        int c = input("c");

        if ((a + b <= c) | (a + c <= b) | (b + c <= a) | (b + a <= c)) error("Треуго

        double P = (a + b + c) / 2;
        return Math.Sqrt(P * (P - a) * (P - b) * (P - c));
    }

    int getFactorial()
    {
        int num = input("факториал");

        int res = 1;
        for (int i = num; i > 1; i--)
            res *= i;
        return res;
    }
```

```
    void error(String message)
    {
        Console.WriteLine("\n"+message+"\n");
        onChoice();
    }

    int input(String title)
    {
        Console.WriteLine("\nВведите " + title);
        Console.Write("> ");
        int num = Convert.ToInt32(Console.ReadLine());
        if (num <= 0) error(title + " не может быть 0 или меньше");
        return num;
    }
    while (true)
    {
        onChoice();
    }
}
```

```
Выбирете действие:
0: поиск площади квадрата
1: поиск площади прямоугольника
2: поиск площади треугольника
3: вычисление факториала числа
4: выход
> 0

Введите a
> 5

площадь квадрата: 25
```

```
Выбирете действие:
0: поиск площади квадрата
1: поиск площади прямоугольника
2: поиск площади треугольника
3: вычисление факториала числа
4: выход
> 1

Введите a
> 3

Введите b
> 5

площадь прямоугольника: 15
```

```
Выбирете действие:
0: поиск площади квадрата
1: поиск площади прямоугольника
2: поиск площади треугольника
3: вычисление факториала числа
4: выход
> 2

Введите a
> 4

Введите b
> 4

Введите c
> 4

площадь треугольника: 6,92820323027551
```

```
Выбирете действие:
0: поиск площади квадрата
1: поиск площади прямоугольника
2: поиск площади треугольника
3: вычисление факториала числа
4: выход
> 3

Введите факториал
> 5

фактриал = 120
```

Контрольные вопросы:

1. Какие сложность бывают у рекурсивных алгоритмов?
    - Простая рекурсия
    - Многократная рекурсия
2. В чём отличие этих сложностей?
    - в колличестве итераций