

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно -
вычислительных систем (КИБЭВС)

РЕКУРСИЯ. ТИПЫ РЕКУРСИЙ

Отчет по практической работе №4
по дисциплине «Языки программирования»

Выполнил

Студент гр. 728-2

_____ Д. Р. Геворгян

____.11.2019

Принял

Преподаватель кафедры КИБЭВС

_____ М. А. Кальнеус
оценка подпись

____.11.2019

1 Введение

Цель работы: целью работы является изучение различных типов рекурсии и дальнейшее их применение для решения практических задач.

Язык программирования C#.

2 Ход работы

2.1 Теоретические сведения

Существует несколько типов рекурсий:

1. Линейная — рекурсивные вызовы на любом рекурсивном срезе инициируют не более одного последующего рекурсивного вызова;
2. Повторительная — частный случай линейной, в которой отсутствуют предварительные или отложенные вычисления;
3. Каскадная — не является линейной, рекурсивные вызовы могут возникнуть более одного раза;
4. Удалённая — в теле функции при рекурсивных вызовах в качестве параметров снова встречаются рекурсивные вызовы этой функции;
5. Косвенная (взаимная) — циклическая последовательность вызовов нескольких функций друг друга.

В данной работе для каждого типа рекурсии будет написана функция, реализующая конкретный тип для решения задачи.

2.2 Написание программы

Линейная рекурсия реализована в алгоритме извлечения целой части от деления a на b путём вычитания меньшего числа из большего.

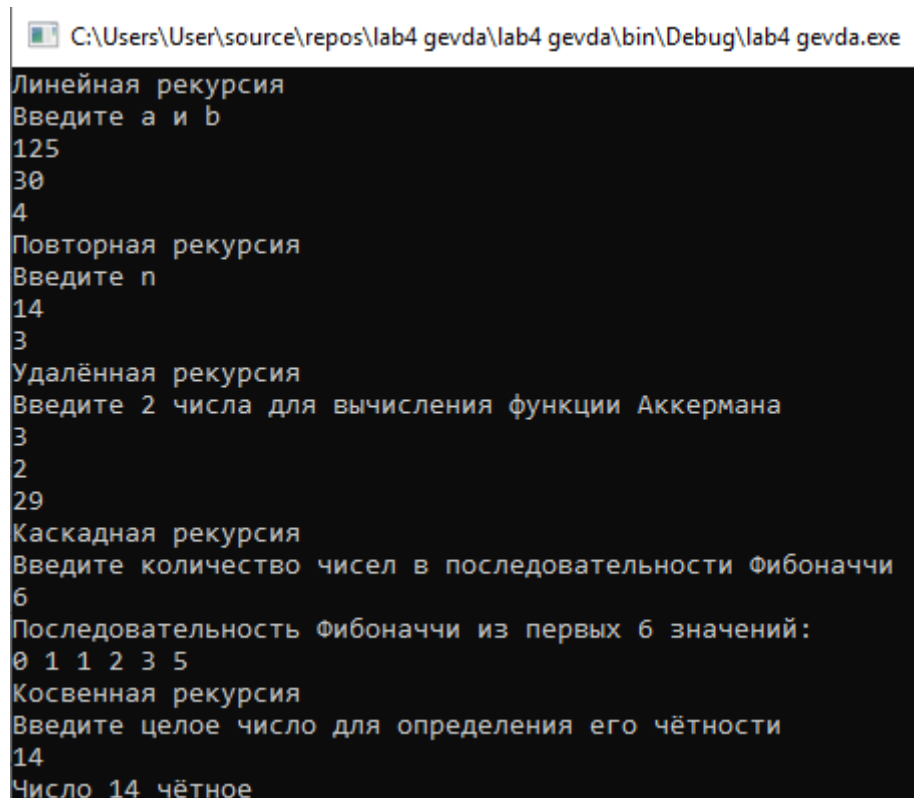
Повторительная рекурсия реализована в алгоритме подсчёта количества цифр 1 в двоичном представлении числа n .

Каскадная рекурсия используется в алгоритме для нахождения первых n членов последовательности чисел Фибоначчи. Сумма n -ого члена находится как сумма функции от $n-1$ и функции от $n-2$ членов.

Косвенная рекурсия используется в функциях проверки числа на чётность и нечётность. Вызывается одна из функций, куда на вход подаётся число, затем функции вызывают друг друга, уменьшая число на 1 до тех пор, пока число не будет равно 0.

Удалённая рекурсия используется в алгоритме вычисления значения функции Аккермана. На вход поступают два целых числа, затем функция вызывает саму себя до тех пор, пока первое число не будет равно 0 или второе число не будет больше 0, а первое не будет равно 0.

Результаты работы функций изображены на рисунке 2.1. Исходный код программы находится в приложении А.



```
C:\Users\User\source\repos\lab4 gevda\lab4 gevda\bin\Debug\lab4 gevda.exe
Линейная рекурсия
Введите a и b
125
30
4
Повторная рекурсия
Введите n
14
3
Удалённая рекурсия
Введите 2 числа для вычисления функции Аккермана
3
2
29
Каскадная рекурсия
Введите количество чисел в последовательности Фибоначчи
6
Последовательность Фибоначчи из первых 6 значений:
0 1 1 2 3 5
Косвенная рекурсия
Введите целое число для определения его чётности
14
Число 14 чётное
```

Рисунок 2.1 – Результаты работы рекурсивных функций

3 Заключение

В ходе выполнения практической работы были изучены типы рекурсии, получены практические навыки по применению различных типов рекурсии для решения практических задач, получены навыки программирования на C#

Отчёт был написан согласно ОС ТУСУР 2013.

Приложение А
(обязательное)

```
using System;

namespace lab4_gevda
{
    class Program
    {
        public static int linear(int a, int b)
        {
            if (a < b)
                return 0;
            else return (linear(a - b, b) + 1);
        }

        public static int repeatable(int n)
        {
            if (n == 0)
                return 0;
            else if (n % 2 == 0)
                return (repeatable(n / 2));
            else return (repeatable((n - 1) / 2) + 1);
        }

        public static bool Even(int m)
        {
            if (m == 0)
                return true;
            else return Odd(m - 1);
        }

        public static bool Odd(int m)
        {
            if (m == 0)
                return false;
            else return Even(m - 1);
        }

        public static int fibonacci(int f)
        {
            if (f == 0)
                return 0;
            else if (f == 1)
                return 1;
            else return (fibonacci(f - 1) + fibonacci(f - 2));
        }

        public static int ackermann(int x, int y)
        {
            if (x == 0)
                return (y + 1);
            else if (y == 0)
                return ackermann(x - 1, 1);
            else return ackermann(x - 1, ackermann(x, y - 1));
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Линейная рекурсия\nВведите а и b");
        }
    }
}
```

```

int a = int.Parse(Console.ReadLine());
int b = int.Parse(Console.ReadLine());
Console.WriteLine($"{linear(a, b)}");
Console.WriteLine("Повторная рекурсия\nВведите n");
int rep = int.Parse(Console.ReadLine());
Console.WriteLine($"{repeatable(rep)}");
Console.WriteLine("Удалённая рекурсия\nВведите 2 числа для вычисления функции
Аккермана");
int ack1 = int.Parse(Console.ReadLine());
int ack2 = int.Parse(Console.ReadLine());
Console.WriteLine($"{ackermann(ack1, ack2)}");
Console.WriteLine("Каскадная рекурсия\nВведите количество чисел в
последовательности Фибоначчи");
int fib = int.Parse(Console.ReadLine());
Console.WriteLine($"Последовательность Фибоначчи из первых {fib} значений:");
for (int i = 0; i < fib; i++)
    Console.Write($"{fibonacci(i)} ");
Console.WriteLine();
Console.WriteLine("Косвенная рекурсия\nВведите целое число для определения его
чётности");
int num = int.Parse(Console.ReadLine());
if (Even(num) == true)
    Console.WriteLine($"Число {num} чётное");
else Console.WriteLine($"Число {num} нечётное");
Console.ReadKey();
    }
}
}

```