Задание 1.

* Параметризация: Используем параметр n, который представляет текущее число в последовательности.
* База рекурсии: Если n ≤ 0, то выводим 0 и завершаем рекурсию.
* Декомпозиция: На каждом шаге печатаем n и вызываем функцию с n - 2.

program recursiveSequence;

procedure printSequence(n: integer);

begin

writeln(n); // print the current number

if n > 0 then // base case: sop recursion when n ≤ 0

printSequence(n - 2); // decomposition: decrease n by 2 and call itself

end;

begin

printSequence(25); // start the recursive call from 25

end.

Задание 2.

* Параметризация: параметр n, который указывает текущее число, для которого вычисляется сумма.
* База рекурсии: если n = 1, возвращаем 1, так как sumTo(1) = 1.
* Декомпозиция: sumTo(n) = n + sumTo(n - 1), уменьшаем n и вызываем функцию снова.

program recursiveSum;

function sumTo(n: integer): integer;

begin

if n = 1 then // base case: if n = 1, return 1

sumTo := 1

else // decomposition: sumTo(n) = n + sumTo(n - 1)

sumTo := n + sumTo(n - 1);

end;

begin

writeln(sumTo(5)); // example call: sumTo(5) = 5 + 4 + 3 + 2 + 1 = 15

end.

Дерево рекурсии:

sumTo(5)

sumTo(4)

sumTo(3)

sumTo(2)

sumTo(1) \*\*(Base case, returns 1)\*\*

Задание 3.

* Параметризация: a (основание) и b (показатель степени).
* База рекурсии: если b = 0, то a^0 = 1.
* Декомпозиция: a^b = a \* a^(b-1), уменьшаем b и вызываем функцию снова.

program recursivePower;

var x, y: integer;

function power(a, b: integer): integer;

begin

if b = 0 then // base case: any number to the power of 0 is 1

power:= 1

else // decomposition: a^b = a \* a^(b-1)

power := a \* power(a, b - 1);

end;

begin

writeln('число:');

readln(x);

writeln('степень:');

readln(y);

writeln(power(x, y)); // output the result of x^y

end.

Задание 4.

* Параметризация: i (текущий шаг) и n (общее количество шагов).
* База рекурсии: если i > n, то рекурсия останавливается.
* Декомпозиция: печатаем привет и i, затем вызываем LoopFor(i + 1, n).

program recursiveLoop;

procedure loopFor(i, n: integer);

begin

if i > n then // base case: stop recursion when i > n

exit;

writeln('hello ', i); // print greeting and counter

loopFor(i + 1, n); // decomposition: increase i and call the function again

end;

begin

loopFor(1, 10); // start recursion from 1 to 10

end.

Задание 5.

* Параметризация: два числа a и b, для которых ищем НОД.
* База рекурсии: если b = 0, то a — это НОД.
* Декомпозиция: вызываем gcd(b, a mod b), заменяя a на b, а b на остаток от деления.

program recursiveGCD;

function gcd(a, b: integer): integer;

begin

if b = 0 then // base case: if remainder is 0, return a as GCD

gcd := a

else // decomposition: gcd(a, b) = gcd(b, a mod b)

gcd := gcd(b, a mod b);

end;

begin

writeln(gcd(3430, 1365)); // example: find GCD of 3430 and 1365

end.

Задание 6.

* Параметризация: i (предыдущее число) и n (текущее число).
* База рекурсии: если число Фибоначчи достигло 10 итераций, рекурсия прекращается.
* Декомпозиция: вычисляем следующее число как i + n и вызываем fib(n, i + n).

program recursiveFibonacci;

procedure fib(i, n, count: integer);

begin

if count > 10 then // base case: stop after 10 numbers

exit;

writeln(n, ' '); // print the current Fibonacci number

fib(n, i + n, count + 1); // decomposition: move to the next Fibonacci number

end;

begin

fib(0, 1, 1); // start recursion from fib(0,1)

end.