**Задание 1.**

START

READ number n

IF n==1 THEN return

FOR i = 1, i <= n, j+1

print “\*”

BREAK

END

Этот алгоритм состоит из трех основных частей: чтение числа n, условия IF и цикла FOR. Давайте рассмотрим временную сложность каждой части по отдельности:

Чтение числа n: Эта операция имеет постоянную временную сложность O(1), так как она выполняется независимо от значения числа n.

Условие IF: Это простое условие сравнения n с 1. Оно также имеет постоянную временную сложность O(1), так как выполнение этого условия не зависит от значения n.

Цикл FOR: Цикл выполняется n раз, где n - введенное число. Каждая итерация цикла включает операцию печати символа "\*". Таким образом, общая временная сложность цикла FOR зависит от значения n и равна O(n), где n - введенное число.

Таким образом, общая временная сложность алгоритма, описанного вами, составляет O(n), где n - введенное число. Это означает, что время выполнения алгоритма будет линейно зависеть от значения числа n.

**Задание 2.**

START

READ number n

numbers i = 0, j = 0, a = 0

FOR j = 2, j <=n, j\*2

a = a+ n/2

END

Алгоритм состоит из нескольких шагов: чтение числа n, инициализация переменных i, j и a, и цикла FOR. Рассмотрим временную сложность каждой части по отдельности:

Чтение числа n: Эта операция имеет постоянную временную сложность O(1), так как она выполняется независимо от значения числа n.

Инициализация переменных i, j и a: Эта операция также имеет постоянную временную сложность O(1), так как инициализация переменных выполняется только один раз.

Цикл FOR: В данном случае, цикл выполняется log(n) раз, где log обозначает логарифм по основанию 2. Каждая итерация цикла включает операцию сложения a и n/2. Таким образом, общая временная сложность цикла FOR зависит от значения n и равна O(log n).

Таким образом, общая временная сложность этого алгоритма составляет O(log n), где n - введенное число. Это означает, что время выполнения алгоритма будет зависеть от логарифма от значения числа n, что является более эффективным, чем линейная зависимость.

**Задание 3.**

START

READ number n

number a = 0

FOR i = 0, i<n, i+1

FOR j=n, j>i, j-1

a = a+ i+j

END

Временная сложность алгоритма можно определить, рассмотрев количество операций, выполняемых в зависимости от размера входных данных.

В данном случае у нас два вложенных цикла. Первый цикл выполняется n раз, а второй цикл, вложенный в первый, выполняется (n - i) раз для каждого значения i в первом цикле.

Таким образом, общее количество операций можно выразить следующим образом:

n + (n - 1) + (n - 2) + ... + 1

Это арифметическая прогрессия суммы, которую можно записать как (n \* (n + 1)) / 2.

Таким образом, временная сложность алгоритма равна O(n^2), где n - входной параметр.

**Задание 4.**

START

READ number n

numbers a = 0, i=n

WHILE i>0

a = a+ i

i=i/2

END

Временная сложность этого алгоритма можно определить, рассмотрев количество операций, выполняемых в зависимости от размера входных данных.

В данном случае, у нас есть цикл while, который будет выполняться до тех пор, пока значение переменной i больше 0. В каждой итерации цикла происходит уменьшение значения i в два раза (i = i/2) и выполнение операции сложения (a = a + i).

Предположим, что изначальное значение i равно n. Цикл будет выполняться, пока i > 0. Заметим, что i будет делиться на 2 на каждой итерации, пока не достигнет значения 0 или меньше. Таким образом, цикл будет выполняться примерно log2(n) раз.

В каждой итерации цикла выполняется операция сложения (a = a + i). Таким образом, общее количество операций сложения будет пропорционально количеству итераций цикла, то есть log2(n).

Таким образом, временная сложность алгоритма равна O(log n), где n - входной параметр.