Функция для решения дискретного логарифма методом Полларда

function pollard_rho_log(p::Int, a::Int, b::Int) # Определяем порядок элемента a по модулю p function order(a::Int, p::Int)::Int r = 1 while mod(a^r, p) != 1 r += 1 end return r end

```
r = order(a, p)  # Порядок a
println("Порядок a: $r")

# Случайное отображение f
function f(c::Int, log_c::Int, x::Int, r::Int)::Tuple{Int, Int}
    if c < r ÷ 2
        return (mod(a * c, p), mod(log_c + 1, r))
    else
        return (mod(b * c, p), mod(log_c + x, r))
    end
end

# Инициализация
u, v = rand(1:p-1), rand(1:p-1)
c, log_c = mod(a^u * b^v, p), mod(u + v, r)
d, log_d = c, log_c

# Итерации
for i in 1:10^6  # Ограничение на количество итераций
    c, log_c = f(c, log_c, 1, r)
    d, log_d = f(d, log_d, 1, r)
    d, log_d = f(d, log_d, 1, r)

    if c == d
        # Решаем уравнение log_a(c) ≡ log_a(d) (mod r)
        x = mod(log_d - log_c, r)
        return x
    end
end

return nothing  # Если решение не найдено
```

end

Тестирование

p = 107 a = 10 b = 64

```
x = pollard_rho_log(p, a, b) if x !== nothing println("Решение: x = $x") else println("Решение не найдено.") end
```