# Лабораторная работа №6: Презентация.

## Разложение чисел на множители.

Евдокимов Максим Михайлович. Группа - НФИмд-01-24.[1]

03 октября, 2024, Москва, Россия

[1]Российский Университет Дружбы Народов

# Цели и задачи работы

# Цель лабораторной работы

Изучить представленные способы разложение чисел на множители и реализовать их в коде.

## Задание

Реализовать рассмотренные алгоритмы программно (алгоритм реализующий p-метод Полларда).

ρ-метод Полларда (метод ρ-Полларда) — это вероятностный алгоритм факторизации целых чисел, который находит нетривиальный делитель числа $n$ за время, пропорциональное $\sqrt[4]{n}$.

Алгоритм основан на поиске цикла в псевдослучайной последовательности чисел, получаемой с помощью некоторой функции $f(x)$, примененной к числу $x$ по модулю $n$.

Алгоритм:

1. Выбираем начальное значение $x_0$ и функцию $f(x)$, например, $f(x) = x^2 + 1$.
2. Строим две последовательности чисел:
   - $x_i$ — последовательность, получаемая итеративным применением функции $f(x)$: $x_{i+1} = f(x_i) \mod n$.
   - $y_i$ — последовательность, получаемая с задержкой: $y_{i+1} = f(f(y_i)) \mod n$.
3. Вычисляем наибольший общий делитель НОД$(|x_i - y_i|, n)$ на каждом шаге.
4. Если НОД $\neq 1$ и НОД $\neq n$, то мы нашли нетривиальный делитель $n$.
5. Если НОД $= n$, то алгоритм завершился неудачно, и нужно выбрать другое начальное значение $x_0$ или функцию $f(x)$.

Особенности:

- **Эффективность:** В среднем алгоритм работает за время $O(n^{1/4})$, что значительно быстрее полного перебора делителей.
- **Вероятностный:** Алгоритм не гарантирует нахождение делителя, и в худшем случае может работать долго.
- **Простота реализации:** Алгоритм легко реализуется и требует небольшого объема памяти.
- **Применимость:** Метод хорошо подходит для факторизации чисел среднего размера (до 100 десятичных цифр).

**Пример:**

Рассмотрим число $n = 8051$.

1. Выбираем $x_0 = 2$ и $f(x) = x^2 + 1$.
2. Строим последовательности:
   - $x_1 = f(2) = 5$
   - $y_1 = f(f(2)) = 26$
3. Вычисляем НОД$\left(|5 - 26|, 8051\right) = 97$.
4. 97 — нетривиальный делитель 8051.

**Заключение:**

Р-метод Полларда — это эффективный и простой в реализации алгоритм факторизации, который широко используется в криптографии и других областях.

# Ход работы

# Функция p-метод Полларда

```julia
# функция реализующая p-метод Полларда для нахождения нетривиального делителя числа n.
using Primes

function pollard_rho(n::Integer, max_iter::Integer=5000) # больше количество итераций нет смысла.
    x, y, d, iter, c = 2, 2, 1, 0, 1
    g = x -> (x^2 + c) % n # функция, используемая в методе Полларда.

    while d == 1 && iter < max_iter
        x = g(x)
        y = g(g(y))
        d = gcd(abs(x - y), n)
        iter += 1
    end

    if d == n
        return nothing  # Фактор не найден
    else
        return d
    end
end
```

```
pollard_rho (generic function with 2 methods)
```

**Рис. 1:** Функция p-метод Полларда и библиотека для простых чисел

# Функция разложения и проверки числа n

```julia
# Функиця для разложения числа n на множители.
function factorize_pollard_rho(n::Integer)
    if isprime(n)
        return "Число n является простым, поэтому фактор не найден."
    end
    factors = []
    while n > 1 # вызывать pollard_rho до тех пор, пока n не будет полностью разложено на простые множители.
        if isprime(n)
            push!(factors, n)
            break
        end
        factor = pollard_rho(n)
        if factor === nothing
            return "Фактор не найден за максимальное число итераций."
        end
        push!(factors, factor)
        n ÷= factor
    end
    return unique(factors) # удаление дубликатов.
end

factorize_pollard_rho (generic function with 1 method)
```

**Рис. 2:** Функция разложения и проверки числа n

```
# Значение для теста
tests = rand(10:10000, 10, 3)
```

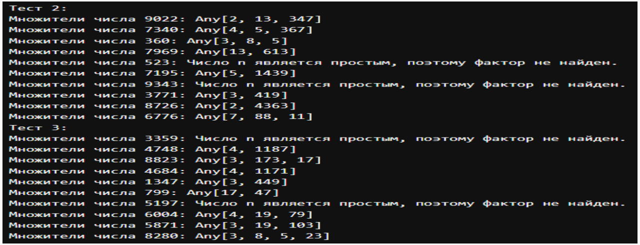
```
10×3 Matrix{Int64}:
 5795  9022  3359
 4653  7340  4748
 3497   360  8823
 4013  7969  4684
 9808   523  1347
 1055  7195   799
 4913  9343  5197
 1755  3771  6004
 2065  8726  5871
 6048  6776  8280
```

**Рис. 3:** Тестовые значения

```julia
# вызов функции, тест.
for i in 1:size(tests)[2]
    println("Тест ", i, ":")
    for j in 1:size(tests)[1]
        factors = factorize_pollard_rho(tests[j, i])
        println("Множители числа ", tests[j, i],": ", factors)
    end
end
```

```
Тест 1:
Множители числа 5795: Any[95, 61]
Множители числа 4653: Any[3, 11, 47]
Множители числа 3497: Any[13, 269]
Множители числа 4013: Число n является простым, поэтому фактор не найден.
Множители числа 9808: Any[16, 613]
Множители числа 1055: Any[5, 211]
Множители числа 4913: Фактор не найден за максимальное число итераций.
Множители числа 1755: Any[3, 5, 13]
Множители числа 2065: Any[7, 5, 59]
Множители числа 6048: Фактор не найден за максимальное число итераций.
```

**Рис. 4:** Вызов функции и тестовая группа 1



```
Тест 2:
Множители числа 9022: Any[2, 13, 347]
Множители числа 7340: Any[4, 5, 367]
Множители числа 360: Any[3, 8, 5]
Множители числа 7969: Any[13, 613]
Множители числа 523: Число n является простым, поэтому фактор не найден.
Множители числа 7195: Any[5, 1439]
Множители числа 9343: Число n является простым, поэтому фактор не найден.
Множители числа 3771: Any[3, 419]
Множители числа 8726: Any[2, 4363]
Множители числа 6776: Any[7, 88, 11]
Тест 3:
Множители числа 3359: Число n является простым, поэтому фактор не найден.
Множители числа 4748: Any[4, 1187]
Множители числа 8823: Any[3, 173, 17]
Множители числа 4684: Any[4, 1171]
Множители числа 1347: Any[3, 449]
Множители числа 799: Any[17, 47]
Множители числа 5197: Число n является простым, поэтому фактор не найден.
Множители числа 6004: Any[4, 19, 79]
Множители числа 5871: Any[3, 19, 103]
Множители числа 8280: Any[3, 8, 5, 23]
```

**Рис. 5:** тестовая группа 2 и тестовая группа 3

# Выводы по проделанной работе

В ходе выполнения лабораторной работы я изучил способ нахождение всех множителей числа алгоритмом реализующим p-метод Полларда.