# Лабораторная работа №2: Шифры перестановки"

### Дисциплина: Математические основы защиты информации и информационной безопасности"

***Савченко Елизавета Николаевна, НПИмд-01-24, 1132249569*** **Российский университет дружбы народов, Москва, Россия**

27 сентября 2025

## Цель работы

Ознакомиться с классическими примерами шифров перестановки.

## Задание

1. Реализовать шифры, представленные в задании.

# Теоретическое введение

## Виды шифров

Шифры подразделяются на:

- Симметричные;
- Асимметричные.

## Виды симметричных шифров

Среди симметричных шифров выделяют:

- Шифры перестановки;
- Шифры подстановки.

# Выполнение лабораторной работы

## Реализация маршрутного шифрования (1)

```
julia> function route_encrypt(text::AbstractString, rows::Int)
           len = length(text)
               cols = ceil(Int, len / rows)
           padded_len = rows * cols
           padded_text = rpad(text, padded_len, ' ')

           arr = collect(padded_text)

               # reshape возвращает матрицу cols x rows,
           # поэтому меняем на permutedims для транспонирования
               mat = permutedims(reshape(arr, cols, rows))

                   encrypted = join(mat[:])
                       return encrypted
                       end
route_encrypt (generic function with 1 method)
```

## Реализация маршрутного шифрования (2)

```
julia> function route_decrypt(cipher::AbstractString, rows::Int)
           len = length(cipher)
               cols = len ÷ rows

           arr = collect(cipher)

               mat = reshape(arr, rows, cols)
                   decrypted_mat = permutedims(mat)

                       decrypted = join(decrypted_mat[:])
                           return strip(decrypted)
                           end
route_decrypt (generic function with 1 method)

julia>

julia> text = "Пример маршрутного шифрования"
"Пример маршрутного шифрования"

julia> rows = 5
5

julia>

julia> encrypted = route_encrypt(text, rows)
"П у врмтшаианинмрофиешгряррoo "

julia> println("Зашифрованный текст: ", encrypted)
Зашифрованный текст: П у врмтшаианинмрофиешгряррoo

julia>

julia> decrypted = route_decrypt(encrypted, rows)
"Пример маршрутного шифрования"

julia> println("Расшифрованный текст: ", decrypted)
Расшифрованный текст: Пример маршрутного шифрования
```

**Результат работы кода для маршрутного шифрования**

```
julia> function route_decrypt(cipher::AbstractString, rows::Int)
           len = length(cipher)
               cols = len ÷ rows

           arr = collect(cipher)

               mat = reshape(arr, rows, cols)
                   decrypted_mat = permutedims(mat)

                       decrypted = join(decrypted_mat[:])
                           return strip(decrypted)
                           end
route_decrypt (generic function with 1 method)

julia>

julia> text = "Пример маршрутного шифрования"
"Пример маршрутного шифрования"

julia> rows = 5
5

julia>

julia> encrypted = route_encrypt(text, rows)
"П у врмтшаианинмрофиешгряррoo "

julia> println("Зашифрованный текст: ", encrypted)
Зашифрованный текст: П у врмтшаианинмрофиешгряррoo

julia>

julia> decrypted = route_decrypt(encrypted, rows)
"Пример маршрутного шифрования"

julia> println("Расшифрованный текст: ", decrypted)
Расшифрованный текст: Пример маршрутного шифрования
```

## Реализация шифрования с помощью решёток (1.1)

```
julia> function rotate_left90(A::Array{Bool,2}, k::Int=1)
           for _ in 1:k
               A = permutedims(A, (2,1))[:, end:-1:1]
                   end
           return A
       end
rotate_left90 (generic function with 2 methods)

julia>
```

## Реализация шифрования с помощью решёток (1.2)

```
julia> function grille_encrypt(text::AbstractString, grille::Array{Bool,2})
          n, m = size(grille)
              total_cells = n * m

          chars = collect(text)  # массив символов

          # Если текста меньше, дополним пробелами
          if length(chars) < total_cells
                  append!(chars, [' ' for _ in 1:(total_cells - length(chars))])
                      else
              chars = chars[1:total_cells]
                  end

          mat = Array{Char}(undef, n, m)
          fill!(mat, ' ')

          idx = 1
          for rot in 0:3
              current_mask = rotate_left90(grille, rot)  # определите функцию поворота
              for i in 1:n, j in 1:m
                  if current_mask[i,j] && mat[i,j] == ' '
                      mat[i,j] = chars[idx]
                                      idx += 1
                  end
              end
          end

          return join(vec(mat))
          end
grille_encrypt (generic function with 1 method)
```

## Результат работы кода для шифрования с помощью решёток

```
julia> function grille_encrypt(text::AbstractString, grille::Array{Bool,2})
           n, m = size(grille)
               total_cells = n * m

           chars = collect(text)  # массив символов

           # Если текста меньше, дополним пробелами
           if length(chars) < total_cells
                   append!(chars, [' ' for _ in 1:(total_cells - length(chars))])
                       else
               chars = chars[1:total_cells]
                   end

           mat = Array{Char}(undef, n, m)
           fill!(mat, ' ')

           idx = 1
           for rot in 0:3
               current_mask = rotate_left90(grille, rot)  # определите функцию поворота
               for i in 1:n, j in 1:m
                   if current_mask[i,j] && mat[i,j] == ' '
                       mat[i,j] = chars[idx]
                                       idx += 1
                   end
               end
           end

           return join(vec(mat))
           end
grille_encrypt (generic function with 1 method)
```

## Реализация таблиц Виженера (1)

```
julia> grille = [
           true  false false false;
               false false true  false;
           false true  false false;
           false false false true
       ]
4×4 Matrix{Bool}:
 1  0  0  0
 0  0  1  0
 0  1  0  0
 0  0  0  1

julia>

julia> text = "Секретноешифрование123"
"Секретноешифрование123"

julia> encrypted = grille_encrypt(text, grille)
"С  о тк  ен е  р"

julia> println("Зашифрованный текст:")
Зашифрованный текст:

julia> println(encrypted)
```

## Реализация таблиц Виженера (2)

```
julia> grille = [
           true  false false false;
               false false true  false;
           false true  false false;
           false false false true
       ]
4×4 Matrix{Bool}:
 1  0  0  0
 0  0  1  0
 0  1  0  0
 0  0  0  1

julia>

julia> text = "Секретноешифрование123"
"Секретноешифрование123"

julia> encrypted = grille_encrypt(text, grille)
"С  о тк  ен е  р"

julia> println("Зашифрованный текст:")
Зашифрованный текст:

julia> println(encrypted)
```

## Результат работы кода для таблиц Виженера

```
julia> grille = [
           true  false false false;
               false false true  false;
           false true  false false;
           false false false true
       ]
4×4 Matrix{Bool}:
 1  0  0  0
 0  0  1  0
 0  1  0  0
 0  0  0  1

julia>

julia> text = "Секретноешифрование123"
"Секретноешифрование123"

julia> encrypted = grille_encrypt(text, grille)
"С  о тк  ен е  р"

julia> println("Зашифрованный текст:")
Зашифрованный текст:

julia> println(encrypted)
```

# Выводы по проделанной работе

## Вывод

В результате работы мы ознакомились с традиционными моноалфавитными шрифтами простой замены, а именно:

- Маршрутным шифрованием;
- Шифрованием с помощью решёток;
- Таблицами Виженера.

Были записаны скринкасты:

- выполнения лабораторной работы;
- создания отчёта по результатам выполения лабораторной работы;

- создания презентации по результатам выполнения лабораторной работы;
- защиты лабораторной работы.