ЛАБОРАТОРНАЯ РАБОТА №2

Шифры перестановки

Кузнецов Юрий Владимирович

# Введение

В данном отчёте будет представлена реализация шифров перестановки

# Основное содержание

## Шифры простой замены:

* Маршрутное шифрование
* Шифрование с помощью решеток
* Таблица Виженера

# Кодовая реализация

## Маршрутное шифрование

alph = [  
 "а", "б", "в", "г", "д", "е", "ж", "з", "и", "к",   
 "л", "м", "н", "о", "п", "р", "с", "т", "у", "ф",   
 "х", "ц", "ч", "ш", "щ", "ъ", "ы", "ь", "э", "ю", "я"  
]  
  
str = "нельзя недооценивать противника"  
password = "пароль"  
dimension = (m = 5, n = 6)  
  
function matrixify(str, dimension)  
 matr = replace(str, " " => "")   
 matr\_chars = collect(matr)   
 res = Vector{Vector{Char}}(undef, dimension.m)   
 counter = 1  
  
 for i in 1:dimension.m  
 res[i] = matr\_chars[counter:min(counter + dimension.n - 1, end)]  
 counter += dimension.n  
 end  
  
 return res  
end

## Маршрутное шифрование

function matrixFill(matrix, dimension)  
   
 while length(matrix[end]) < dimension.n  
 push!(matrix[end], 'а')  
 end  
  
 return matrix  
end  
  
function pushindex(str, alph)  
 indexes = Int[]  
 for ch in collect(str)   
 index = findfirst(==(string(ch)), alph)   
 if index !== nothing  
 push!(indexes, index)  
 end  
 end  
 return indexes  
end  
  
function result(arr, indexes, dimension)  
 result\_str = ""  
 map = sort([Dict(:item => indexes[i], :key => i) for i in 1:length(indexes)],   
 by = x -> x[:item])  
  
 print(map)  
  
 for i in 1:dimension.n  
 counter = map[i][:key]  
 for j in 1:dimension.m  
 char = arr[j][counter]  
 result\_str \*= string(char)   
 end  
 end  
  
 return uppercase(result\_str)  
end

## Маршрутное шифрование

matrix = matrixFill(matrixify(str, dimension), dimension)  
indexes = pushindex(password, alph)  
result\_str = result(matrix, indexes, dimension)  
  
println(result\_str)  
end

## Шифрование с помощью решеток

function rot90(matrix::Array{T,2}) where {T}  
 return [matrix[j, end-i+1] for i in 1:size(matrix, 1), j in 1:size(matrix, 2)]  
end  
  
function encrypt\_with\_rails(input\_text::String, cipher\_key::String, matrix\_dim::Int)  
 base\_matrix = fill("", matrix\_dim, matrix\_dim)  
 encryption\_matrix = fill("", 2 \* matrix\_dim, 2 \* matrix\_dim)  
 char\_sequence = 1  
 result\_text = ""  
 sanitized\_text = replace(input\_text, " " => "")  
  
 for i in 1:matrix\_dim, j in 1:matrix\_dim  
 base\_matrix[i, j] = string(char\_sequence)  
 encryption\_matrix[i, j] = string(char\_sequence)  
 char\_sequence += 1  
 end  
  
 for row in 1:(2\*matrix\_dim)  
 for col in (2\*matrix\_dim):-1:1  
 if encryption\_matrix[row, col] == ""  
 base\_matrix = rot90(base\_matrix)  
 for i in 0:(matrix\_dim-1), j in 0:(matrix\_dim-1)  
 encryption\_matrix[row+i, col-j] = base\_matrix[matrix\_dim-i, matrix\_dim-j]  
 end  
 end  
 end  
 end

## Шифрование с помощью решеток

char\_sequence = 1  
 used\_positions = String[]  
 for char in sanitized\_text  
 placed = false  
 for i in 1:(2\*matrix\_dim), j in 1:(2\*matrix\_dim)  
 if encryption\_matrix[i, j] == string(char\_sequence) && !placed  
 position\_key = string(i, ",", j)  
 if !(position\_key in used\_positions)  
 encryption\_matrix[i, j] = string(char)  
 push!(used\_positions, position\_key)  
 placed = true  
 end  
 end  
 end  
 char\_sequence += 1  
 if char\_sequence > matrix\_dim^2  
 char\_sequence = 1  
 empty!(used\_positions)  
 end  
 end

## Шифрование с помощью решеток

sorted\_key = sort(collect(cipher\_key))  
 for key\_char in sorted\_key  
 column\_index = findfirst(==(key\_char), cipher\_key)  
 for row in 1:(2\*matrix\_dim)  
 cell = encryption\_matrix[row, column\_index]  
 result\_text \*= cell != "" ? cell : " "  
 end  
 end  
  
 return result\_text  
end  
  
text="word"  
key="key"  
matrix\_size = 2  
  
println(encrypt\_with\_rails(text, key, matrix\_size))

## Таблица Виженера

function cipher\_vigenere(msg::String, secret::String)  
 alpha\_range = 'a':'z'  
 result\_text = ""  
 pos\_in\_key = 1  
  
 for char in msg  
 if isletter(char)  
 shift = findfirst(==(secret[pos\_in\_key]), alpha\_range) - 1  
 char\_pos = findfirst(==(char), alpha\_range) + shift  
 char\_pos > 26 && (char\_pos -= 26)  
 result\_text \*= alpha\_range[char\_pos]  
 pos\_in\_key += 1  
 pos\_in\_key > length(secret) && (pos\_in\_key = 1)  
 else  
 result\_text \*= char  
 end  
 end  
  
 return result\_text  
end  
  
msg="hello"  
secret="key"  
  
println(cipher\_vigenere(msg, secret))

# Заключение

В данной лабораторной работе были реализованы шифры перестановки (Маршрутное шифрование, Шифрование с помощью решеток, Таблица Виженера)