```
---
title: "Лабораторная работа №2"
format:
  html:
    toc: true
    code-fold: true
    code-line-numbers: true
    code-tools: true
---


## Тема: Шифры перестановки

**Выполнила:** Исламова Сания Маратовна (студ. билет 1132249576)

---

```{julia}
#| label: cipher-program
#| fig-cap: "Реализация шифров Цезаря и Атбаш"
#| code-line-numbers: "true"
#| warning: false
#| eval: false



# Маршрутное шифрование


function columnar_main()

    # Бесконечный цикл для работы программы до команды выхода

    while true

        # Выводим меню с доступными командами

        println("Введите Ш для шифрования, Р для расшифрования, В для выхода")

        # Приглашение для ввода команды
        print(">>> ")
        # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру
        cmd = lowercase(strip(readline()))

        # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
        cmd == "в" && (println("Выход"); break)
        # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и продолжаем цикл
        cmd in ["ш", "р"] || (println("Неверная команда"); continue)

        # Запрашиваем текст для шифрования/расшифрования
        print("Введите текст: ")
        # Читаем введенный текст
        text = readline()
        # Запрашиваем пароль для шифрования
        print("Введите пароль: ")
        # Читаем введенный пароль
        password = readline()

        # ПОДГОТОВКА ТЕКСТА:
```
```

```
        # Удаляем пробелы и приводим к верхнему регистру
        clean_text = replace(uppercase(text), " " => "")
        # Преобразуем пароль в массив символов (для избежания проблем с индексацией русских символов)
        pass_chars = collect(uppercase(password))
        # n - количество столбцов (равно длине пароля)
        # m - количество строк (вычисляем округлением вверх длины текста / длины пароля)
        n, m = length(pass_chars), ceil(Int, length(clean_text) / length(pass_chars))

        # СОЗДАНИЕ ТАБЛИЦЫ:
        # Дополняем текст символами 'А' до полного заполнения таблицы m×n
        padded = clean_text * "А"^(m*n - length(clean_text))
        # Преобразуем строку в массив символов и reshape в матрицу m×n
        table = reshape(collect(padded), (m, n))

        # СОРТИРОВКА СТОЛБЦОВ:
        # Создаем пары (символ пароля, индекс столбца) для каждого столбца
        column_pairs = [(pass_chars[i], i) for i in 1:length(pass_chars)]
        # Сортируем пары по символам пароля (алфавитный порядок)
        sort!(column_pairs, by = x -> x[1])
        # Извлекаем отсортированные индексы столбцов
        sorted_cols = [idx for (char, idx) in column_pairs]

        # ФОРМИРОВАНИЕ РЕЗУЛЬТАТА:
        # Выписываем символы из таблицы по столбцам в новом порядке
        # Сначала все строки первого столбца, затем второго и т.д.
        result = join([table[i,j] for j in sorted_cols for i in 1:m])

        # Выводим результат шифрования
        println("Результат: $result")
        # Разделительная линия для визуального отделения
        println("-"^50)
    end
end

#Запуск основной функции программы
columnar_main()




# Шифрование с помощью решёток


function fleissner_main()

    # Бесконечный цикл для работы программы до команды выхода
    while true

        # Выводим меню с доступными командами
        println("Введите Ш для шифрования, Р для расшифрования, В для выхода")

        # Приглашение для ввода команды
        print(">>> ")

        # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру
        cmd = lowercase(strip(readline()))
```

```
117
118         # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
119         cmd == "в" && (println("Выход"); break)
120         # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и
    продолжаем цикл
121         cmd in ["ш", "р"] || (println("Неверная команда"); continue)
122
123         # Запрашиваем текст для шифрования
124         print("Введите текст: ")
125         # Читаем введенный текст
126         text = readline()
127         # Запрашиваем пароль (должен содержать 4 символа для решетки 2x2)
128         print("Введите пароль (4 символа): ")
129         # Читаем введенный пароль
130         password = readline()
131
132         # ПОДГОТОВКА ТЕКСТА:
133         # Удаляем пробелы, приводим к верхнему регистру и преобразуем в массив
    символов
134         clean_chars = collect(replace(uppercase(text), " " => ""))
135         # Преобразуем пароль в массив символов
136         pass_chars = collect(uppercase(password))
137         # k = 2 означает решетку 2x2, которая создает маску 4x4
138         k = 2  # размер решетки
139
140         # СОЗДАНИЕ РЕШЕТКИ 4x4:
141         # Размер большой решетки (2k × 2k = 4x4)
142         size_2k = 2k
143         # Создаем булеву маску (false - закрыто, true - прорезь)
144         grille = falses(size_2k, size_2k)
145         # Заполняем маску прорезями в 4 угловых квадратах 2x2
146         for i in 1:k, j in 1:k
147             grille[i, j] = grille[i, k+j] = grille[k+i, j] = grille[k+i, k+j] = true
148         end
149
150         # ЗАПОЛНЕНИЕ ТАБЛИЦЫ:
151         # Общее количество ячеек в решетке
152         total = size_2k^2
153         # Если текст короче, дополняем символами 'A'
154         length(clean_chars) < total && append!(clean_chars, fill('A', total -
    length(clean_chars)))
155         # Инициализируем таблицу для заполнения, индекс текста и копию маски
156         table, idx, mask = fill(' ', size_2k, size_2k), 1, copy(grille)
157
158         # ЧЕТЫРЕ ЭТАПА ЗАПОЛНЕНИЯ (0°, 90°, 180°, 270°):
159         for _ in 1:4
160             # Проходим по всем ячейкам решетки
161             for i in 1:size_2k, j in 1:size_2k
162                 # Если ячейка - прорезь и есть еще символы для записи
163                 mask[i,j] && idx <= length(clean_chars) && (table[i,j] =
    clean_chars[idx]; idx += 1)
164             end
165             # Поворачиваем маску на 90° по часовой стрелке для следующего этапа
166             mask = reverse(mask, dims=1)'
167         end
168
169         # ФОРМИРОВАНИЕ РЕЗУЛЬТАТА:
170         # Сортируем индексы столбцов по алфавитному порядку символов пароля
171         sorted_cols = sort(1:length(pass_chars), by=i -> pass_chars[i])
172         # Выписываем символы из таблицы по столбцам в новом порядке
```

```
        result = join([table[i,j] for j in sorted_cols for i in 1:size_2k])

        # Выводим результат шифрования
        println("Результат: $result")
        # Разделительная линия для визуального отделения
        println("-"^50)
    end
end

#Запуск основной функции программы
fleissner_main()




# Таблица Виженера


function vigenere_main()

    # Создаем русский алфавит как массив символов для корректной работы с индексами

    alphabet = collect("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ")

    # Сохраняем длину алфавита для модульных вычислений

    n = length(alphabet)

    # Бесконечный цикл для работы программы до команды выхода
    while true
        # Выводим меню с доступными командами
        println("Введите Ш для шифрования, Р для расшифрования, В для выхода")
        # Приглашение для ввода команды
        print(">>> ")
        # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру
        cmd = lowercase(strip(readline()))

        # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
        cmd == "в" && (println("Выход"); break)
        # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и
продолжаем цикл
        cmd in ["ш", "р"] || (println("Неверная команда"); continue)

        # Запрашиваем текст для шифрования/расшифрования
        print("Введите текст: ")
        # Читаем введенный текст
        text = readline()
        # Запрашиваем пароль для шифрования
        print("Введите пароль: ")
        # Читаем введенный пароль
        password = readline()

        # ПОДГОТОВКА ТЕКСТА:
        # Удаляем пробелы, приводим к верхнему регистру и преобразуем в массив
символов
        clean_chars = collect(replace(uppercase(text), " " => ""))
        # Аналогично обрабатываем пароль
```

```
        pass_chars = collect(replace(uppercase(password), " " => ""))

        # СОЗДАНИЕ ПОВТОРЯЮЩЕГОСЯ КЛЮЧА:
        # Создаем пустой массив символов для ключа
        key_chars = Char[]
        # Для каждого символа текста определяем соответствующий символ ключа
        for i in 1:length(clean_chars)
            # Циклически повторяем пароль: (i-1) % length + 1 дает циклический индекс
            push!(key_chars, pass_chars[(i-1) % length(pass_chars) + 1])
        end

        # ШИФРОВАНИЕ/ДЕШИФРОВАНИЕ:
        # Создаем массив для результата
        result_chars = Char[]
        # Обрабатываем каждый символ текста
        for i in 1:length(clean_chars)
            text_char = clean_chars[i]    # Текущий символ текста
            key_char = key_chars[i]       # Соответствующий символ ключа

            # Находим позиции символов в алфавите
            text_idx = findfirst(==(text_char), alphabet)
            key_idx = findfirst(==(key_char), alphabet)

            # Если оба символа найдены в алфавите
            if text_idx !== nothing && key_idx !== nothing
                if cmd == "ш"
                    # ШИФРОВАНИЕ: (текст + ключ) mod n
                    new_idx = (text_idx + key_idx - 1) % n
                    # Обработка случая, когда mod дает 0
                    new_idx == 0 && (new_idx = n)
                else
                    # ДЕШИФРОВАНИЕ: (текст - ключ) mod n
                    new_idx = (text_idx - key_idx) % n
                    # Обработка отрицательных результатов
                    new_idx <= 0 && (new_idx += n)
                end
                # Добавляем преобразованный символ к результату
                push!(result_chars, alphabet[new_idx])
            else
                # Если символ не из алфавита, добавляем как есть (пробелы, знаки препинания)
                push!(result_chars, text_char)
            end
        end

        # Преобразуем массив символов обратно в строку
        result = String(result_chars)
        # Выводим результат
        println("Результат: $result")
        # Разделительная линия для визуального отделения
        println("-"^50)
    end
end

#Запуск основной функции программы
vigenere_main()
```